

Informatie.

De μ P Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305. Het gironummer van de vereniging is 3757649.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.
- Het stimuleren en ondersteunen van het gebruik van micro-computers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 5 maal per jaar de μ P Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een softwarebibliotheek en een technisch forum voor dediverse systemen in stand gehouden.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in de μ P Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.

Het telefoonnummer is: 053-303902.

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technische ondersteuning streeft het bestuur ernaar zgn. systeemcoördinatoren te benoemen. Van tijd tot tijd zal in de μ P Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

Het Bestuur:

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Nico de Vries (voorzitter)
Mari Andriessenrade 49
2907 MA Capelle a/d IJssel
Telefoon 010-4517154

Mick Agterberg (secretaris)
Davidvosstraat 29
1063 HV Amsterdam
Telefoon 020-131538

Jacques H.G.M. Banser (penningmeester)
Haaksbergerstraat 199
7513 EM Enschede
Telefoon 053-324137

Gert van Opbroek (Redactie μ P Kenner)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Jan D.J. Derksen
Ed Verkadestraat 9-1
7558 TH Hengelo
Telefoon 074-770970

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ton Smits
De Meren 39
4731 WB Oudebosch

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:
Siep de Vries

Ereleden:
Mevr. H. de Vries van der Winden
Anton Mueller

Inhoud

Colofon

De μ P Kenner

Nummer 64, december '89
Verschijnt 5 maal per jaar
Oplage: 200 stuks
Druk: FEBO Offset, Enschede

De redactie:

Gert van Opbroek
Bram de Bruine
Antoine Megens
Nico de Vries
Joost Voorhaar

Eindredactie:

Gert van Opbroek

Lay-out:

Joost Voorhaar

Redactieadres:

Gert van Opbroek,
Bateweg 60
2481 AN Woubrugge

Vereniging

Informatie.....	2
Redactioneel.....	4
Uitnodiging voor de clubbijeenkomst.....	5
Verslag van de algemene ledenvergadering.....	6
Taakverdeling bestuursleden.....	28
Van de voorzitter.....	29

Algemeen

Dikhuidige Personeelsindeling.....	30
Errata.....	36
Een verhaal betreffende bulletin borden.....	37
De HCC-dagen.....	38
Figuren.....	43

DOS-65

Windows onder DOS65 met een Z80 kaart.....	9
PAUSE en LABEL, twee MS-DOS bekenden.....	18
Virtual Eprom Disk voor DOS65.....	23

Hardware

Transputers: bouwstenen van de toekomst.....	7
Computers (Deel 5).....	32

MS-DOS

De IBM-PC en z'n klonen (Deel 6).....	39
---------------------------------------	----

Talen/Software

Babbage: De Taal van de Toekomst.....	15
---------------------------------------	----

De μ P Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μ P Kenner verschijnt vijf maal per jaar, in principe op de derde zaterdag van de maanden februari, april, juni, augustus, oktober en december.

Copy voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze copy kan in papier-, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Copy kan ook op het Bulletin Board van de vereniging gepost worden in de redactie area. Nadere informatie kan bij het redactieadres of via het bulletin board opgevraagd worden.

De redactie houdt zich het recht voor copy zonder voorafgaand bericht niet of slechts gedeeltelijk te plaatsen of te wijzigen. Geplaatste artikelen blijven het eigendom van de auteur en mogen niet zonder diens voorafgaande schriftelijke toestemming door derden geplubliceerd worden, in welke vorm dan ook.

De redactie noch het bestuur kan verantwoordelijk gesteld worden voor toepassing(en) van de geplaatste copy.

Redactioneel.

Veranderingen (1):

Het zal u wel opgevallen zijn dat deze uitgave van de uP Kenner er iets anders uit ziet als de voorgaande. Dat wordt veroorzaakt door het feit dat we vanaf deze uitgave het blad met een professioneel DTP (Desktop Publishing) -pakket) gaan opmaken (lay-outen in vaktermen). Er heeft zich een vrijwilliger (Joost Voorhaar) gemeld die dat graag voor ons wil gaan doen. Jacques Banser ondersteunt hem hierbij en ik ben er van overtuigd dat er een zeer fraai blad uit de bus zal komen.

De vorige uitgave is binnen de club door Ernst Elderenbosch gedrukt en niet door de drukkerij die de afgelopen twee jaar het blad voor ons gedrukt heeft. Dit was naar aanleiding van het feit dat we de kosten van drukken van het blad wat wilden reduceren. Met alle respect voor en dank aan Ernst en degenen die hem hierbij geholpen hebben willen we toch een poging doen de kwaliteit van het drukwerk tegen acceptabele kosten wat beter te maken en zodoende wordt dit blad weer door een andere (professionele) drukker gedrukt. In de nabije toekomst wordt ook nog de omslag van het blad veranderd maar dat ziet u in het februarinummer wel.

Vanwege het feit dat de opmaak van het blad langs elektronische weg gedaan wordt, is het wel noodzakelijk dat alle kopij in magnetische vorm beschikbaar is. Dit betekent dat voordat Joost en Jacques met de opmaak beginnen er eerst voor gezorgd moet worden dat de kopij door een MS-DOS machine gelezen kan worden. Om deze reden wil ik u vragen alle artikelen, programma's etc. op een floppy of cassette aan mij op te sturen. Het redactie-team is in staat de volgende formaten te lezen:

- Alle disketteformaten van MS-DOS doch bij voorkeur 5¹/₄ inch, 360 kB
- Diskettes voor de AMIGA
- Diskettes voor de Atari ST
- 5¹/₄ inch diskettes voor DOS-65 (alle formaten)
- Diskettes voor EC-65(k)
- Basicode II
- Cassettes voor de Junior
- Etc., etc.....

U ziet, mogelijkheden te over. Verder is het zo dat als u iets wilt publiceren en u weet niet of wij in staat zijn iets met uw diskette-formaat te doen, dan kunt u het beste even contact met mij opnemen. In overleg met u en met mijn contactpersonen zullen we dan gezamenlijk een oplossing zoeken. Kopij op papier mag natuurlijk ook, alleen houdt dat in dat uw redacteur het over zal moeten tikken en dat kost tijd en inspanning.

Het gevolg van de veranderingen in het blad is wel dat ook ik er aan moet geloven WordPerfect te leren. Tot nu toe kon ik me nog aardig redden met Wordstar (versie 3.2) maar om aan te sluiten bij de rest van de tekstverwerk-wereld, ben ik toch maar met WP begonnen.

Veranderingen (2):

Behalve de groep mensen die het blad maken is ook het bestuur van de vereniging gewijzigd. Er zijn een aantal mensen afgetreden en er zijn een aantal nieuwe mensen in het bestuur gekomen. In een apart verhaaltje wordt de nieuwe samenstelling van het bestuur beschreven. Op deze plaats wil ik de oud bestuursleden heel hartelijk danken voor de prettige samenwerking en voor de inspanningen die ze zich voor de club getroost hebben. Verder wil ik de nieuwe bestuursleden in de eerste plaats feliciteren met hun verkiezing en veel geluk toewensen bij het besturen van de KIM Gebruikersclub Nederland. Als ik naar de samenstelling van het bestuur kijk ben ik er van overtuigd dat we een bestuur gekregen hebben dat de club weer nieuw leven in kan blazen. Verder is het nu zo dat binnen het bestuur zowel DOS-65, EC-65(k), AMIGA, ATARI ST en MS-DOS vertegenwoordigd zijn; kortom het bestuur vormt een waarschijnlijk wat dat betreft wel een redelijke afspiegeling van de vereniging.

December:

Het is weer december en dat betekent dat we weer te maken hebben met de feestdagen. Deze Kenner zult u ook wel temidden van een stapel gelukwensen voor Kerst en het oud en nieuw op uw deurmat of in uw brievenbus gevonden hebben. Uiteraard wil het bestuur en de redactie u ook langs deze weg hele prettige kerstdagen en een heel goed 1990 toewensen. Ik hoop dat u de komende jaren, samen met de KIM Gebruikersclub Nederland, nog heel veel plezier aan uw computershobby zult beleven. Verder zou ik het, als redacteur, prettig vinden als u mij ook wilt helpen bij het samenstellen van een jaargang prima uP Kenners door mij zo nu en dan van kopij te voorzien. Alleen met hulp van de leden kunnen we de KIM Gebruikersclub Nederland voor de leden behouden.

Uw Redacteur:

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: Zaterdag 20 januari 1990
 Locatie: Nieuwe kantine FORBO-Krommenie
 Industrieweg 12
 1566 JP Assendelft
 Telefoon: 075-291911

Entree: f10, = voor het gedeelte na 11:00 uur

Routebeschrijving

Per auto:

1. Uit de richting Amsterdam: Coentunnel door en de Coentunnelweg helemaal afrijden. Aan het einde rechtsaf (water aan de linkerzijde). Dan de 1e afslag rechtsaf, richting Uitgeest-Alkmaar. Doorrijden tot aan de stoplichten. Linksaf de spoorbaan over.

2. Na 75 meter linksaf: Industrieweg. Links aanhoudende komt men dan op het FORBO-terrein.

3. Uit de richting Alkmaar: Snelweg Alkmaar-Haarlem. Afslag Uitgeest-Zaandam. Bij kruising linksaf. Bij de 3e stoplicht ten rechtsaf, de spoorbaan over. Verder volgens punt 2.

Per trein:

Station Krommenie-Assendelft. Rechtsaf tot over de spoorwegovergang. Zie verder punt 2.

Programma:

9:30 Zaal open. Ontvangst met koffie
 10:00 Opening door de voorzitter en verwelkoming door de gastheer: Co Filmer.
 10:10 Diashow over FORBO-Krommenie.
 10:30 Ledenvergadering

Agenda:

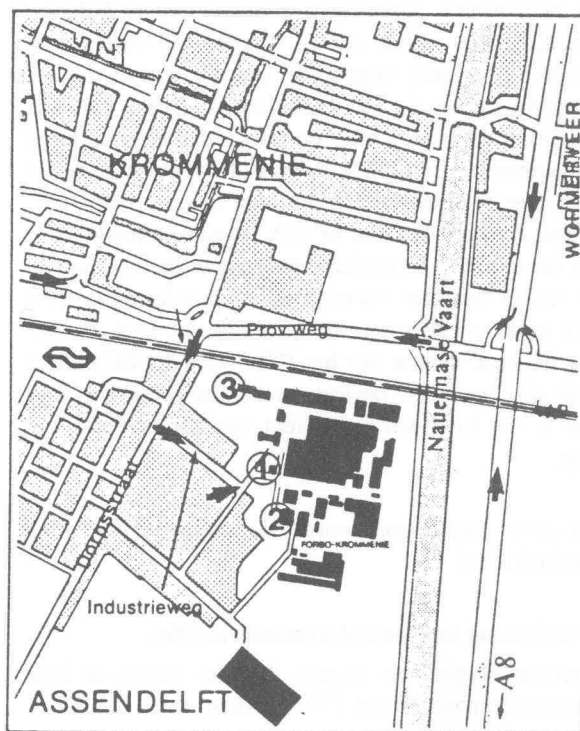
- Opening
- Notulen vorige vergadering
- Behandeling jaarverslag 1989
- WVTTK
- Rondvraag
- 10:45 Koffiepauze.
- 11:00 Voordracht van onze gastheer Co Filmer over machinesturingen met behulp van PLC's en micro-processors.
- 12:00 Forum en markt.
- 12:30 Lunch, aangeboden door FORBO- Krommenie.

Aansluitend het informele gedeelte bedoeld om kennis, ervaring en Public Domain software uit te wisselen. Breng daarom ook uw systeem mee.

17:00 Sluiting.

Attentie

Het is ten strengste verboden illegale kopieën te verspreiden. Aan personen die deze regel overtreden, zal de verdere toegang tot de bijeenkomst ontzegd worden. Breng alleen software mee die u legaal in uw bezit heeft. Het bestuur aanvaardt geen enkele aansprakelijkheid voor de gevolgen van het in bezit hebben van illegale software.



- | | | |
|---|--|---|
| ① Portier
Portier
Portier
Gateman
Portero | ② Ontvangstcentrum
Salle de reception
Empfangsraum
Reception building
Sala de recepcion | ③ Kantoor
Bureaux
Buros
Offices
Oficinas |
| ④ Centraal magazijn
Magasin central
Zentrallager
Central warehouse
Almacen central | | |
- Treinverb. Amsterdam-Alkmaar (half uurdienst)**
 Chemin de fer Amsterdam-Alkmaar (toutes les demi-heures)
 Eisenbahn Amsterdam-Alkmaar (jede halbe Stunde)
 Railway Amsterdam-Alkmaar (half hour service)
 Linea ferroviaria Amsterdam-Alkmaar (cada media hora)

Verslag van de algemene ledenvergadering van 11 nov 1989

1. Opening.

De voorzitter opent de vergadering en constateert dat de opkomst laag is.

2. Notulen van de buitengewone vergadering van 07 okt 1989.

In het verslag wordt ten onrechte vermeld dat Ernst Elderenbosch kandidaat zou zijn voor een bestuursfunctie. Wel is hij bereid om copy aan te leveren voor de μ P Kenner. Met deze opmerking wordt het verslag goedgekeurd.

3. Concept begroting 1990.

De begroting is gebaseerd op een verwacht ledental van 120. Met name op de drukkosten van de μ P Kenner is in deze begroting flink bezuinigd. De afschrijvingen op de inventaris zijn volgend jaar iets lager dan dit jaar. Onder de post ledenwerving moet bijvoorbeeld verstaan worden dat wij via een ander toch nog een presentatie kunnen doen op de HCC dagen. Verder worden er ook kosten gemaakt voor het verzenden van reclame materiaal. De kosten van de Sysop bestaan voornamelijk uit telefoonkosten. Het blijkt overigens in de praktijk dat veel niet-leden gebruik maken van het BBS. Van de 400 bellers over de afgelopen periode waren er 60 leden. Helaas is het ledenwervingseffect van het BBS vrij gering.

Na deze toelichting wordt de concept begroting goedgekeurd.

4. verkiezing kascontrolecommissie 1990.

Antoine Megens en Jaques Prenger zullen de kascontrolecommissie van 1990 vormen.

5. Verkiezing nieuwe bestuursleden.

Zoals reeds aangekondigd zullen een drietal bestuursleden hun functies neerleggen.

Het zijn:

Rinus Vleesch Dubois (voorzitter).

Adri Hankel (lid).

Gert klein (secretaris).

De volgende leden hebben zich kandidaat gesteld voor een bestuursfunctie:

Mick Agterberg, kandidaat voorzitter. Mick beheert een aantal zaken in Amsterdam en lijkt daarmee een goede kandidaat voor de voorzittersfunctie.

Geert Stappers, is vooral geïnteresseerd in hardware.

Ton Smits, is al een aantal jaren in de picture als voortrekker van de EC-65.

De vergadering gaat akkoord met de verkiezing van deze kandidaten in het bestuur. Het nieuwe bestuur zal onderling uitmaken hoe de bestuurstaken verdeeld zullen worden.

Rinus, Adri en Gert krijgen als dank voor hun jarenlange inspanningen voor de club een (vlocibare = drinkbare) attentie aangeboden. Voor Rinus Vleesch Dubois die al sinds de oprichting van de club in 1977 in het bestuur zit is er nog een extra kado: de club KIM die door Rinus zelf in een koffer is ingebouwd en een flinke nostalgische waarde vertegenwoordigd.

6. Rondvraag.

Er zijn geen vragen.

Wageningen, 11 nov 1989

Gert Klein

TRANSPUTERS: BOUWSTENEN VAN DE TOEKOMST.

In de computerwereld rommelt het. Deskundigen voorspellen dat het einde aan de absolute hegemonie van Motorola en Intel op de microprocessor markt niet lang meer op zich zal laten wachten. "RISC" luidt hier het toverwoord. "RISC" is een afkorting van "Reduced Instruction Set Computer", een term die al aangeeft dat het hier gaat om processoren die slechts een beperkt aantal instructies kennen. Deze instructies zijn dan echter wel bijzonder krachtig en... razendsnel! Een heel bekende computer die op RISC-technologie gebaseerd is, is de Archimedes. Hier zit een speciaal door Acorn vervaardigde RISC-processor in die het mogelijk maakt dat een programma in BASIC dat op een Archimedes werkt, vaak net zo snel of zelfs sneller is dan een vergelijkbaar programma in bijv. de taal "C" dat op een PC loopt.

Een andere fabrikant die hoge ogen scoort in de RISC technologie is de INMOS corporation. Dit bedrijf werd in 1978 opgericht met het doel VLSI schakelingen te ontwikkelen en te fabriceren. Twee jaar later, in 1980 dus, kwamen de eerste bruikbare geheugen-IC's uit de ovens van het bedrijf. De directeur van de ontwikkelingsafdeling, Iann Barron, liep echter al sinds 1975 met plannen rond voor het maken van een RISC-achtige processor. Het zou echter 10 jaar duren voordat in 1985 zijn geesteskind, de transputer zoals hij het ding gedoopt had, in de handel kwam.

De transputer is een processor die zich onderscheidt van de meeste andere processoren doordat het ding een viertal "links" heeft. Via deze links kan de transputer razendsnel communiceren met andere transputers, wat hem uitermate geschikt maakt voor toepassingen in multi-processor systemen. Ze kunnen op velerlei manieren in een netwerk opgenomen worden. De vorm van een transputernetwerk staat niet vast. Een transputernetwerk kan volgens allerlei verschillende structuren geconfigureerd zijn. Typische voorbeelden van dergelijke structuren zijn o.a. de boomstructuur, de roosterstructuur en de thoroïde roosterstructuur. Van dit rijtje is met name de

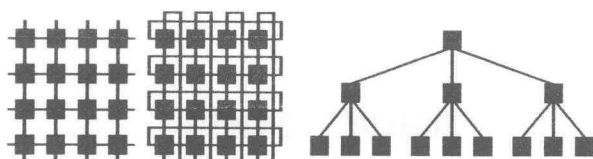


Fig. 1: Rooster-, Thoroïde- en boom-structuur

boomstructuur erg populair. Ter illustratie kan je je een flightsimulator voorstellen. Als wortel van de boom dient een transputer die alleen maar taken staat te verdelen. Deze transputer heeft drie "bladeren": een transputer zorgt voor de beeldinformatie, eenje zorgt voor realistische reacties van het instrumentenpaneel en de derde verwerkt de positie van de stuurknuppel en pedalen die door de piloot bedient worden. De transputer die voor de beeldverwerking zorgt zou op zich ook weer een aantal "bladeren" kunnen hebben. Bijvoorbeeld één transputer die zich bezig houdt met de positionering van de achtergrond, en een andere die de details voor z'n rekening neemt.

Als voorbeeld van een transputer heb ik in deze artikelenreeks de T414 van INMOS Co. genomen. Dit is een 32-bits RISC-processor met 2 kByte statisch RAM aan boord, een viertal links en, wat minstens net zo belangrijk is, een hardware matige process-scheduler. De laatste zorgt er voor dat de transputer in een recordtijd tussen processen kan wisselen. Eén van de belangrijkste andere uit de transputerserie van INMOS is de T800. Het enige verschil met de hier behandelde T414 is het de numerieke co-processor die bij de T800 op de chip geïntegreerd is.

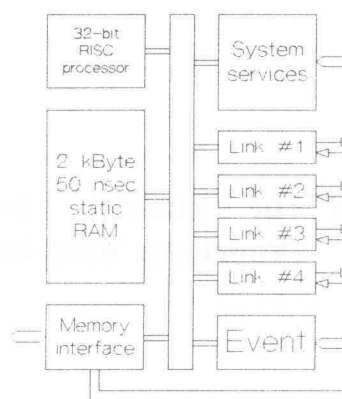


Fig. 2: De T414 van INMOS

Laten we bij het begin beginnen. Als hart van deze bouwstenen fungeert dus een 32-bits RISC processor. Deze processor werkt volgen het normale Von Neumann principe: instructie ophalen, decoderen, aanvullende informatie ophalen en tenslotte voert 'ie 't zaakje nog uit ook. Deze processor is voorzien van slechts een zestal registers: de instruction-pointer, een zg. workspace-pointer, een "operand-register" en een drietal algemene registers die samen de rekenstack vormen. Er zijn nog een aantal andere pointers die van belang zijn voor het werken van de transputer, maar deze zullen behandeld worden als het gaat over process-switching, in een van de andere artikelen van deze reeks. Het operand register

doet dienst bij het samenstellen van de instructie. Daar de transputer instructies kent ter grootte van een nibble zowel als instructies die 32 bits in beslag nemen is hier een speciaal register voor gereserveerd. De workspace pointer wijst in het werkgeheugen van de transputer dat als algemeen toepasbare registerset opgevat kan worden. De process-scheduler hoeft zo alleen deze twee pointers aan te passen om een ander process te activeren.

Wat de transputer echt onderscheidt van de meeste andere processoren zijn de links. Een link is een seriële verbinding via welke de transputer snel kan communiceren met de buitenwereld. Hij kan zelfs over één van de links worden geboot, waardoor het systeem heel gecontroleerd opgestart kan worden. De hostcomputer kan bijvoorbeeld gestart worden van ROM. Als het host-systeem draait gaat deze de eerste transputer booten. Is dit eenmaal gebeurt, dan boot de tweede vanuit de eerste transputer, de derde vanuit de tweede etc. Als de laatste transputer draait is het systeem klaar voor gebruik.

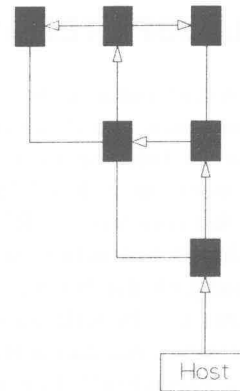
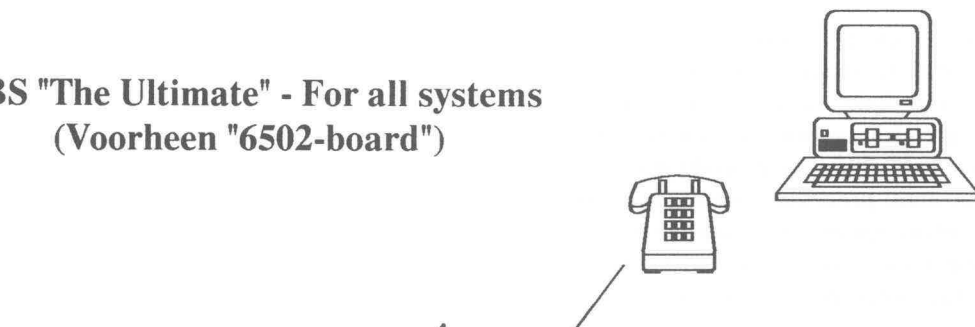


Fig. 3: Bootproces van een multi-transputer systeem

In de volgende afleveringen ga ik dieper in op de fysieke aspecten van de transputer, laat zien hoe de process-switching geregeld is en ook een globaal overzicht van de instructionset zal de revue passeren. Tenslotte volgen dan in de laatste aflevering nog een aantal voorbeelden van toegepaste systemen.

J. Voorhaar.

**BBS "The Ultimate" - For all systems
(Voorheen "6502-board")**



Tel: 053 - 303902

Windows onder DOS65 met een Z80 kaart

In het vorige artikel heb ik de window subroutines beschreven voor een standaard dos65 machine. Ik beloofde toen een set subroutines die gebruik kunnen maken van de Z80 kaart van elektuur om een scherm in op te bergen. Deze subroutines maken gebruik van exact dezelfde adressen en parameters als de routines van de vorige keer, dus je kunt eenvoudigweg deze routines laden in \$e400-\$e5ff ipv. de andere routines zonder wijzigingen aan te moeten brengen in je programma's. Er moet een kleine kanttekening bij geplaatst worden: De elektuur Z80 kaart 'zit' normaal op

\$e300-\$e303. Dit vond ik niet praktisch want het waren de enige adressen in pagina \$exx die gebruikt

werden voor I/O. Daarom heb ik in mijn systeem de Z80 kaart geplaatst op \$e158-\$e15b, vier adressen die goed aansluiten bij de andere I/O adressen. Als je de kaart nu op een ander adres hebt ingesteld, moet je alleen de waarde 'z80bas' (nu \$e258) aan te passen in de source. Er zijn twee listings, de source 'windowz.mac' (met de z van Z80), en de file graphlib.mac, waarin alle graphics characters vermeld staan met hun resp. hexcodes. Zo kun je de routines gemakkelijker aanpassen als je een andere character generator gebruikt.

succes,

Ernst Elderenbosch

```

; file          graphlib.mac
;
; purpose       graphic character definitions
; author        Ernst Elderenbosch
; system        dos65
; date          18 september 1988
;
; io65 standard graphics
;
vb      equ      $18      ; vertical bar
hb      equ      $19      ; horizontal bar
tl      equ      $10      ; top left corner
tr      equ      $16      ; top right corner
bl      equ      $12      ; bottom left corner
br      equ      $14      ; bottom right corner
tm      equ      $17      ; top bar with middle connection
lm      equ      $11      ; left bar "
bm      equ      $13      ; bottom bar "
rm      equ      $15      ; right bar "
vh      equ      $1a      ; crossed vertical and horizontal bar
;
; file          windowz.mac
;
; purpose       allow windowing on dos65 machines
;               using a z80 second processor card at $e158-e15b
; author        E.R.Elderenbosch
; date          08 sept 1988
;
;
; lib          graphlib
; file          graphlib.mac
;
; purpose       graphic character definitions
; author        Ernst Elderenbosch
; system        dos65
; date          18 sept 1988
;

```

; io65 standard graphics

```

;
vb      equ      $18      ; vertical bar
hb      equ      $19      ; horizontal bar
tl      equ      $10      ; top left corner
tr      equ      $16      ; top right corner
bl      equ      $12      ; bottom left corner
br      equ      $14      ; bottom right corner
tm      equ      $17      ; top bar with middle connection
lm      equ      $11      ; left bar ,,
bm      equ      $13      ; bottom bar ,,
rm      equ      $15      ; right bar ,,
vh      equ      $1a      ; crossed vertical and horizontal bar
;
pointr  equ      $0000     ; memory address pointer [2]
points  equ      $0002     ; memory address pointer save [2]
txtadr  equ      $0004     ; memory address text pointer [2]
;
z80bas  equ      $e158     ; z80 base address
;
prtext  equ      $c03b     ; print string till $0
;
;      org      $e400      ; free memory (2 pages)
;
;      init
;
savewi  jmp      copy1     ; copy screen area to scratch memory
makewi  jmp      border    ; make window border & clear text area
fillwi  jmp      instxt    ; copy text to text area
remowi  jmp      copy2     ; restore original text from scratch memory
;
crdxy1  res      2         ; x,y coordinates top left border
crdxy2  res      2         ; x,y coordinates bottom right border
txtvec  res      2         ; memory address text to be placed in window
adrxy1  res      2         ; screen memory address top left
adrxy2  res      2         ; screen memory address bottom right
linlen  res      2         ; hor & vert length of window
lincnt  res      1         ; line count, temp variable
;
copy1   jsr      iniz80    ; init z80 interface pia
        lda      #1        ; copy to z80 command
        jsr      outz80
        lda      #$10      ; dest address $1000
        jsr      outz80
        lda      #0
        jsr      outz80
        lda      #8        ; copy 2k bytes = $0800
        jsr      outz80
        lda      #0
        jsr      outz80
;
1       lda      #0
        sta      pointr
        lda      #$e8
        sta      pointr + 1
        lda      #$f0      ; end of screen memory area
        sta      points + 1
2       ldy      #0
        lda      [pointr],y

```



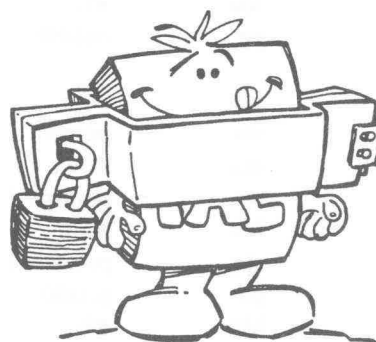
```

        jsr      outz80
        inc      pointr
        bne      2.b
        inc      pointr + 1
        lda      pointr + 1
        cmp      points + 1
        bne      2.b
        rts

;
border  jsr      gtmadr          ; get memory address of coordinates
        jsr      intadr         ; initialise address pointers
;
top     ldy      #0              ; subroutine does check screen wrap.
        ldx      #0
        lda      #tl
        sta      [pointr],y
1       jsr      updptx          ; increment pointer
        inx
        cpx      linlen
        beq      2.f
        lda      #hb
        sta      [pointr],y
        bra      1.b
2       lda      #tr
        sta      [pointr],y
;
sides   lda      linlen + 1
        sta      lincnt
1       dec      lincnt
        jsr      updpty
        ldy      #0
        ldx      #0
        lda      lincnt
        beq      bottom
;
2       lda      #vb
        sta      [pointr],y
3       jsr      updptx
        inx
        cpx      linlen
        beq      4.f
        lda      #$20
        sta      [pointr],y
        bra      3.b
4       lda      #vb
        sta      [pointr],y
        bra      1.b
;
bottom  lda      #bl
        sta      [pointr],y
1       jsr      updptx
        inx
        cpx      linlen
        beq      2.f
        lda      #hb
        sta      [pointr],y
        bra      1.b
2       lda      #br
        sta      [pointr],y

```

Secure Data



```

                rts
;
instxt          jsr      gtmadr          ; get memory address of coordinates
                jsr      intadr          ; initialise address pointers
                jsr      updpty          ; first blank line
                jsr      updptx          ; first char pos
                lda      txtvec          ; get txt address
                sta      txtadr
                lda      txtvec + 1
                sta      txtadr + 1
                ldx      #1
1              ldy      #0
                lda      [txtadr],y
                pha
                jsr      inctxt
                pla
                beq      9.f              ; 0 = end of window text
                cmp      #$d
                beq      2.f              ; next line within window
                sta      [pointr],y
                jsr      updptx
                inx
                cpx      linlen
                bne      1.b
2              jsr      updpty
                jsr      updptx
                ldx      #1
                bra      1.b
9              rts
;
copy2           lda      #3              ; copy from z80 command
                jsr      outz80
                lda      #$10            ; org address $1000
                jsr      outz80
                lda      #0
                jsr      outz80
                lda      #8              ; copy 2k bytes = $0800
                jsr      outz80
                lda      #0
                jsr      outz80
;
1              lda      #0
                sta      pointr
                lda      #$e8
                sta      pointr + 1
                lda      #$f0            ; end of screen memory area
                sta      points + 1
2              jsr      getz80
                ldy      #0
                sta      [pointr],y
                inc      pointr
                lda      pointr
                bne      2.b
                inc      pointr + 1
                lda      pointr + 1
                cmp      points + 1
                bne      2.b
                lda      #$ff
                jsr      outz80

```

```

;
;gtmadr    sec
;          lda    crdxy2          ; calculate hor length
;          sbc    crdxy1          ; if carry cleared, = error x2 x1
;          sta    linlen
;          inc    linlen          ; take care of border
;          lda    crdxy2 + 1
;          sbc    crdxy1 + 1
;          sta    linlen + 1
;          inc    linlen + 1
;
; 1        ldx    crdxy2          ; get end x and y
;          ldz    crdxy2 + 1
;          jsr    $f024          ; position cursor
;          lda    $e724          ; get memory address current position
;          sta    adrx2          ; and save it
;          lda    $e724 + 1
;          sta    adrx2 + 1
;          jsr    $f027          ; restore cursor position
;
; 2        ldx    crdxy1          ; get start x and y
;          ldz    crdxy1 + 1
;          jsr    $f024          ; position cursor
;          lda    $e724          ; get memory address current position
;          sta    adrx1          ; and save it
;          lda    $e724 + 1
;          sta    adrx1 + 1
;          jsr    $f027          ; restore cursor position ?
;          rts
;
;intadr    lda    adrx1          ; initialise pointers
;          sta    pointr
;          sta    points
;          lda    adrx1 + 1
;          sta    pointr + 1
;          sta    points + 1
;          rts
;
;updptx    inc    pointr          ; update pointer (increment by one and check
;          lda    pointr          ;           for screen wrap)
;          bne    1.f
;          inc    pointr + 1
;          lda    pointr + 1
;          cmp    #$f0
;          bne    1.f
;          lda    #$e8
;          sta    pointr + 1
;
; 1        rts
;
;updpty    clc                    ; set pointr to next line and check
;          lda    points          ;           for screen wrap
;          adc    #80
;          sta    pointr
;          sta    points
;          lda    points + 1
;          adc    #0
;          cmp    #$f0
;          bne

```



```

1      lda      #$e8
      sta      pointr + 1
      sta      points + 1
      rts

;
inctxt      inc      txtadr
           bne      9.f
           inc      txtadr + 1
9      rts

;
iniz80      lda      #40
           ldy      #44
           ldx      #0
           sta      z80bas + 1
           stx      z80bas
           sty      z80bas + 1
           dex
           sta      z80bas + 3
           stx      z80bas + 2
           sty      z80bas + 3
           ldx      #20
1      lda      z80bas
           dex
           bne      1.b
           rts

;
outz80      pha
1      lda      z80bas + 3
           bpl      1.b
           lda      z80bas + 2
           pla
           sta      z80bas + 2
           rts

;
getz80      1      lda      z80bas + 1
           bpl      1.b
           lda      z80bas
           rts

;
           end      init

```

; set port directions

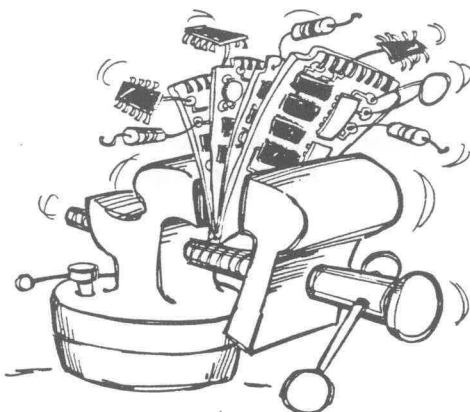
; purge z80 fifo buffer

; wait until last byte accepted
; reset pbc flag

; send byte to z80

; get data byte

Space Efficiency



Babbage: De Taal van de Toekomst.

door: Tim McDonough, Rensselaer Polytechnic Institute

Vertaling: Nico de Vries

(gereproduceerd zonder toestemming, maar even goed: Bedankt!)

Er zijn maar weinig zaken in de computerbranche die meer opwindend zijn dan het ontwerpen van een nieuwe computer taal. De laatste is Ada - het nieuwe superspeeltje van het U.S. Department of Defence. Ada is, zoals u misschien weet, ontworpen om ouderwetse en uit de mode geraakte talen als COBOL en FORTRAN te vervangen.

Het grootste probleem hierbij is, dat zo'n cyclus (ontwerp, gebruik en ouderwets en/of uit de mode raken) twintig tot dertig jaar duurt, terwijl zo'n cyclus pas werkelijk start als we ons beginnen te realiseren dat de huidige talen niet goed meer zijn. Dit proces kan worden kortgesloten door nu al met de opvolger van Ada te beginnen. Dan zal, wanneer we besluiten dat Ada niet meer mee kan, z'n opvolger onmiddellijk beschikbaar zijn.

De nieuwe generatie taal-ontwerpers heeft de gewoonte aangenomen zijn geesteskinderen niet langer al dan niet slimme woordafkortingen als naam mee te geven, maar de taal te noemen naar een persoon die echt bestaan heeft. Pascal is genoemd naar de eerste persoon die een rekenende machine bouwde, en Ada stamt van de naam van eerste computerprogrammeur. De naam van onze taal komt van Charles Babbage, die in armoede stierf terwijl hij probeerde de eerste computer af te bouwen. De nieuwe taal is dus genoemd naar de eerste systeemontwerper die achter was op de planning en ver over het budget was.

Babbage is gebaseerd op taalelement die werden ontdekt nadat Ada voltooid was. C.A.R. Hoare verhaalde van twee manieren om een software-ontwerp te construeren in zijn speech na het winnen van de 1980 ACM Turing Award:

"De eerste manier is het ontwerp zo simpel te maken, dat er duidelijk geen overbodigheden meer in zitten; de tweede manier is om het zo ingewikkeld te maken, dat er geen duidelijke overbodigheden in voorkomen."

De ontwerpers van Babbage hebben een derde manier gekozen - een taal die uitsluitend duidelijke overbodigheden bevat. Programma's geschreven in Babbage zijn zo verschrikkelijk onbetrouwbaar, dat de afdeling onderhoud al de handen vol aan de software heeft voordat de afdeling systeemintegratie het geheel heeft afgerond. Dit levert een perfecte en voortdurende garantie op op een baan in de debug-sector.

Net zoals Pascal gebruikt Ada "Strong Typing" om fouten, ontstaan door het doorelkaar gebruiken van verschillende data-typen te voorkomen. De ontwerpers van Babbage gaven echter de voorkeur aan "Good Typing" om fouten veroorzaakt door spelingsproblemen voortijdig de kop in te drukken. Latere versies van Babbage zullen zelfs "Touch Typing" bezitten, teneinde een behoefte te dekken die al reeds lang werd gevoeld.

Een ander onderwerp van verhitte debatten tussen de verschillen taal-ontwerpers is dat van Parameter Passing naar subfuncties. Sommigen prefereren "Call by Name", terwijl anderen de voorkeur aan "Call by Value" geven. Babbage gebruikt een geheel nieuwe manier "Call by Telephone". Dit is zeer zinvol bij interlokaal "Parameter Passing".

In Ada wordt de nadruk gelegd op "software portability". Babbage daarentegen bevordert "hardware portability", wat heb je

tenslotte aan een computer die je niet mee kunt nemen?

Het is een goed teken wanneer een taal gesponsord wordt door de overheid. COBOL werd bijvoorbeeld door de overheid gesubsidieerd, terwijl Ada met behulp van de fondsen van het U.S. Department of Defence werd ontwikkeld. Na lang onderhandelen werd het Ministerie van (Geestelijke) Volksgezondheid bereid gevonden Babbage te ondersteunen.

Er mogen geen subsets van Ada gebruikt worden. Babbage is juiste het tegenovergestelde hiervan: niets ligt vast, behalve de uitbreidbaarheid ervan. Iedere gebruiker kan/moet zijn eigen versie definiëren en ontwikkelen. Dit maakt meteen een eind aan de discussie of een taal uitgebreid of juist eenvoudig moet zijn: met Babbage kan beide. Babbage is dus de ideale taal voor de 'ik'-generatie. In de onder-

De nieuwe taal is dus genoemd naar de eerste systeemontwerper die achter was op de planning en ver over het budget was

staande voorbeelden krijgt u een idee hoe Babbage eruit ziet. Gestructureerde talen verbanden GOTO's en meervoudige conditionele branches door deze te vervangen de eenvoudige If-Then-Else structuur. Babbage heeft echter een aantal nieuwe conditionele statements die zich als termieten in de structuur van uw programma gedragen:

What if - Wordt gebruikt is simulatietaalen. Deze springt voordat de conditie is ge-evalueerd.

Or Else -Conditionele Bedreiging, zoals in: 'Tel deze twee getallen op, Of Anders...'.

Why Not? Voert de erna volgende code uit met een idee van: 'Wat kan mij het eigenlijk schelen?'.

Who Else? Wordt gebruikt bij polling in I/O-operaties.

Elsewhere Hier is uw programma werkelijk, terwijl u dacht dat het hier was.

Going Going Gone Speciaal voor het schrijven van zeer ongestructureerde programma. Springt naar een willekeurige plek elders in het programma. Minstens zo effectief als 10 GOTO's.

Al jaren gebruiken programmeertalen dingen als "FOR", "DO UNTIL" en "DO WHILE" om dingen te maken die gewoon "LOOP" heten. Teneinde deze trend door te zetten zijn er in Babbage de volgende loop-constructies beschikbaar:

Don't Do While Not Deze loop wordt niet uitgevoerd als de testconditie niet False is (of als het vrijdagmiddag is).

Didn't Do De loop wordt eenmaal uitgevoerd, waarna de sporen die dit achterliet zorgvuldig worden uitgewist.

Can't Do De loop verdwijnt onzichtbaar.

Won't Do De CPU gaat in halt, omdat hij geen trek heeft in de code binnen de loop. De uitvoering van het programma kan hervat worden door op de terminal "MAY I?" in te typen.

Might Do Dit geval hang van het humeur van de processor af. De loop wordt uitgevoerd als de CPU "UP" is, terwijl hij wordt overgeslagen als de CPU "DOWN" is, of als de gevoelens van de CPU werden beledigd.

Do Unto Others Wordt gebruikt om de main loop voor time-sharing systemen te schrijven, zodat alle gebruikers in een uniforme manier tegemoet getreden worden.

Do-Wah Wordt gebruikt om de timing loops in computer gegenereerde muziek te schrijven. Geeft aanleiding tot 'rag-timing'.

Elke zichzelf respecterende gestructureerde taal heeft een CASE-statement om meervoudige branches te kunnen implementeren. ALGOL heeft een geïndexeerd CASE-statement, Pascal doet het met een gelabeld CASE-statement. Niet veel keus dus. Babbage heeft de volgende variëteiten aan boord:

Het Just-in-Case statement Speciaal voor het opnemen van gedachten achteraf en last-minute wijzigingen. Laat zelfs toe om met nul te vermenigvuldigen, teneinde de rampen ontstaan bij het per ongeluk door nul delen weer ongedaan te maken.

Het Brief-Case statement Dit bevordert de ontwikkeling van portable software.

Het Open-and-Shut Case statement Hierbij een geen bevis van juistheid nodig.

Het In-Any-Case statement Deze werkt gewoon altijd.

Het Hopeless-Case statement En deze dus nooit.

Het Basket-Case statement Een werkelijk hopeloos geval.

De Babbage ontwerp-groep is voortdurend bezig nieuwe mogelijkheden te evalueren die kunnen meehelpen te voorkomen dat Babbage-gebruikers überhaupt een niveau van effectiviteit kunnen bereiken. Zo wordt op dit moment het 'Almost Equal'-teken bekeken, dat gebruikt wordt bij het vergelijken van twee glijdende komma getallen. Dit verkleint namelijk de zorg er bijna geweest te zijn.

De Babbage ontwerp-groep is voortdurend bezig nieuwe mogelijkheden te evalueren die kunnen meehelpen te voorkomen dat Babbage-gebruikers überhaupt een niveau van effectiviteit kunnen bereiken

Geen enkele taal, hoe slecht ook, staat op zichzelf. Daarom hebben een werkelijk state-of-the-art operating system nodig om Babbage goed te ondersteunen. Na verschillende commercieel verkrijgbare systemen vergeefs uitgeprobeerd te hebben, besloten we een eigen 'virtueel' operating system te schrijven. Iedereen heeft tegenwoordig virtueel geheugen, waarbij geheugen dat er niet is, op schijf gesimuleerd wordt. Ons nieuwe operating system wordt Virtual Time Operating System (VTOS) genoemd. Terwijl virtuele geheugensystemen het computergeheugen de virtuele grootheid maken, doet VTOS hetzelfde met CPU-tijd.

Het resultaat is dat de computer een oneindig aantal taken tegelijk kan uitvoeren. Net zoals het virtuele geheugen dat om zijn functies te kunnen uitvoeren allerlei geintjes op disk uithaalt, moet VTOS ook enige speciale goocheltrucs uithalen om het gesteld

doel te bereiken. Zo lijkt het misschien dat al uw programma's nu gerund worden, terwijl dat best pas volgende week zou kunnen geschieden.

Zoals gemakkelijk te zien is, staat Babbage nog in de kinderschoenen. DE Babbage ontwerp-groep is dan ook ontvankelijk voor iedere suggestie voor deze krachtige nieuwe taal. Als enig lid van de groep (alle aanvragen voor een lidmaatschap zullen worden gehonoreerd) daag ik de computergebruikers-gemeenschap dan ook uit deze droom bewaarheid te laten worden.

Tenslotte een Babbage programvoorbeeld dat als raamwerk kan dienen voor een ieder in Babbage wil programmeren:

Nico de vries.

```

Program babbage_template
  by Who else
  What if no_program_variables_exist then create_some Or Else!
  Might Do Loop
    Just-In-Case
      program_runs_perfectly : Delete_source + _files;
      user_wants_modifications : Ask_for_a_raise;
      user_wants_demo : Crash_gracefully;
      user_wants_lobotomy : Continue;
    Why Not Do Unto Others until user_panicks?
    Basket-Case
      user_pulls_out_hair : Do_wah until Elsewhere;
      user_pulls_out_terminal : Going Going Gone;
      user_enters_data : Can't Do input_data;
      user_wants_more : append_random_code_to_program_and_recompile;
    Or Else!
  Won't Do Loop
    Windows;
    Diapers;
    Documentation;
  Loop-De-Doop
  Loop-De-Doop.
Margop.

```

Fig. 1: Babbage template, voorbeeld programma

PAUSE en LABEL, twee MS-DOS bekenden

Degenen, die vertrouwd zijn met MS-DOS, zullen de kommando's PAUSE en LABEL wel kennen. Niettegenstaande beide utilities niet echt iets nieuw betekenen voor DOS65, leek het me interessant ze ook voor DOS65 te maken; zij het met enkele verschillen t.o.v. MS-DOS.

PAUSE wordt gebruikt om de verdere afwerking van een command file op bepaalde plaatsen uit te stellen of te onderbreken. Het tussenvoegen van PAUSE in een commandfile stopt de verdere afwerking hiervan, dit om U de tijd te laten bvb. disketten om te wisselen. De commandfile wordt verder afgewerkt door op een willekeurige toets, uitgezonderd Ctrl-C, te drukken.

Als een PAUSE wordt tegengekomen, verschijnt er "strike any key to continue" op het scherm.

Wordt Ctrl-C ingedrukt, dan verschijnt er "Terminate command file ?" op het scherm. Antwoordt men hier bevestigend door Y in te drukken, dan eindigt de command file op die plaats. Elke andere ingave heeft een verdere uitvoering van de command file tot gevolg.

Geeft men na PAUSE een kommentaar, dan verschijnt die in plaats van de "Strike any key to continue".

Luc De Cock

```

; file                pause.mac
;
; program    pause
;
; function    prints a string and waits until a key is pressed
;
; by          De Cock Luc
;
; date        january 89
;
;
;entries
;
sync          equ      $c00f          close all files
input         equ      $c020          get char A
output        equ      $c023          print character in A
crlf          equ      $c02f          print cr & lf
print         equ      $c03b          print string after call
loupch        equ      $c041          convert lower to upper case
sopt          equ      $c068          scan options
clstcon       equ      $ca2a          (re)set consolebit
ermes         equ      $d0b7          error message
;
;zero page addresses
;
t1            equ      $f0
opt           equ      $fe
;
;                org      $a000
;
pause         jmp      pause1
              fcc      $c8,$c5,$cc,$d0
eropt         jsr      print
              fcc      "\rUtility for use in a command file."
              fcc      "\rPrints a string and waits until a key is pressed."
              fcc      "\rDefault the string 'Strike any key to continue' is"
              fcc      " printed."
              fcc      "\rSyntax : PAUSE [-S] [string]"
              fcc      "\rWith Ctrl-C you can end the command file"

```

```

                                fcc      "\rOption : -S : No string is printed."
                                fcc      "\rAbbreviation : PAU\r",0
                                rts
pause1  jsr      sopt
                                fcc      'S',0
                                bcc      1.f
                                jmp      eropt      error : print help text
1       sty      t1
                                sta      t1 + 1
                                txa
                                bmi      wait      -s no string
                                ldy      #$ff
2       iny
                                lda      [t1],y      print string
                                cmp      #'\'r'
                                beq      3.f
                                jsr      output
                                bcc      2.b
3       cpy      #$0
                                bne      wait
                                jsr      print
                                fcc      "\rStrike any key to continue...",0
wait     lda      #$c0      Enable ^C
                                jsr      clstcon
                                jsr      input
                                cmp      #$03      Ctrl-C ?
                                bne      5.f
                                jsr      crlf
                                jsr      print
                                fcc      "\rTerminate command file ? Y/N",0
                                jsr      input      get input
                                jsr      loupch      convert to upper case
                                cmp      #'Y'
                                bne      5.f
                                jsr      sync      if 'Y' close all file
                                jsr      crlf
5       lda      #$c1      Disable ^C
                                jsr      clstcon
                                jsr      crlf
                                rts
;
                                end      pause

```

High Speed



```

; file      label.mac
;
; program   label
;
; author    De Cock Luc
;
; date      april 89
;
; DOS65 entries and functions
;
output      equ      $c023      print character in A
inecho      equ      $c026      get and put character
bufin       equ      $c029      get line to inputbuffer
crlf        equ      $c02f      print cr & lf
hnout       equ      $c035      print A as one or two hex nibbles
print       equ      $c03b      print string after call
loupch      equ      $c041      convert lower to upper case
ermes       equ      $d0b7      error message
readsect    equ      $d0c0      readsector
wrchsect    equ      $d0c9      write and check sector
posit       equ      $f024      position the cursor to X,Y
;
;DOS65 page zero addresses and work memory
;
drive       equ      $86
rwpoin      equ      $e8
curpx       equ      $e704
curpy       equ      $e705
inbuf       equ      $aa00
;
;          org      $a000
;
label       jmp      label1
           fcc      $c8,$c5,$cc,$d0
           fcc      $4c,$4c,$4c
           fcc      "\rCommand to show, change or delete the label (name) of"
           fcc      "\rthe diskette in the chosen drive. Default is drive 1."
           fcc      "\rSyntax : LABEL"
           fcc      "\rNo options"
           fcc      "\rAbbreviation : LA",0
;
label1      jsr      print
           fcc      "\rDrive (0 or 1*)? ",0
           jsr      inecho
           cmp      #$0d
?
           beq      1.f
           cmp      #'0
           beq      2.f
           cmp      #'1
           beq      2.f
           bne      label1
1           lda      #1          default drive
2           and      #3
           sta      drive
;
           jsr      readsis      read system sector
           bcs      errlab
;

```

```

        jsr      print
        fcc      "\rActual label  :",0
        lda      sisblk + $40  read first character
        bne      curlab      not zero : show current
        jsr      print      else : no label
        fcc      "none",0
        jmp      change

;
curlab 1      ldx      #0      print actual label
        lda      sisblk + $40,x
        beq      change
        jsr      output
        inx
        cpx      #$18      end limit name
        bne      1.b

;
change jsr    print
        fcc      "\rChange (Y*/N)? ",0
        jsr      inecho
        cmp      #$0d
        beq      newlab
        jsr      loupch
        cmp      #'Y
        beq      newlab
        jsr      crlf
        rts      exit to dos

;
errlab      jmp      ermes

;
newlab      jsr      print
        fcc      "\rEnter new label : "
        fcc      ".....",0
        ldx      #$13      X position of first point
        ldy      curpy      get current cursorposition Y
        iny      curpy0 = line 1
        jsr      posit      place cursor at first point
        jsr      bufin
        sty      tempc      save length

;
        jsr      readsis      read system sector again :
        bcs      errlab      protection against changing diskettes

;
        ldy      #0
1      cpy      tempc      last input?
        beq      clear      if so clear rest
        lda      inbuf,y      copy label
        sta      sisblk + $40,y into sis
        iny
        cpy      #$17      limit (one 0 needed after label)
        bne      1.b
        beq      writsis

;
clear 2      lda      #0
        sta      sisblk + $40,y clear rest
        iny
        cpy      #$18      total space in sis for label
        bne      2.b

;
        jsr      writsis      write systemsector

```



```

;          bcs          errlab
;
;          rts                      back to dos
;
; readsis   lda          #sisblk&255 set pointer to
;          sta          rwpoin      memory where
;          lda          #sisblk > 8  system sector
;          sta          rwpoin + 1  should come
;          lda          drive
;          ldx          #0           system track
;          ldy          #1           system sector
;          jsr          readsect
;          ora          #$01        flag if error
;          rts
;
; writsis   lda          #sisblk&255 set
;          sta          rwpoin      pointer
;          lda          #sisblk > 1  to
;          sta          rwpoin + 1  memory
;          lda          drive
;          ldx          #0           system track
;          ldy          #1           system sector
;          jsr          wrchsect    write and check
;          ora          #$01        flag if error
;          rts
;
; sisblk    res          256
; tempc     res          1
;
;          end          label

```

A Great Performer!



Virtual Eprom Disk voor DOS65

Waarom na de virtual disk nu nog een epromdisk zullen velen zeggen. Toen ik met DOS65 startte, beschikte ik slechts over 1 diskdrive. Om een beetje opti-maler met DOS65 te kunnen werken, bouwde ik een virtual disk van 512kB (niet volledig identiek aan het schema van de club). Bij het opstarten werden dan de meest gebruikte utilities in een subdirectory hiervan gecopieerd, die dan als systeemdisk fungeerde. Een andere subdirectory werd gereserveerd als workdrive en de root als userdrive. Toen ontstond het idee om de informatie van de systeemdisk in EPROMs te programmeren en deze dan ergens in de beschikbare ruimte, voorzien voor de virtual disk, te adresseren. De systeemdisk zou dan direkt beschikbaar zijn na het booten. De beschikbare ruimte van 1MB zou ik toch nooit volledig gebruiken als ramdisk (reeds met een virtual userdrive van 256k kan men al heel wat uit-richten). Uiteindelijk heb ik gekozen voor virtual epromdisk (v.e.d.) van 256kB (4 * 27512), hierin kan men heel wat utilities en eventuele programmeer-talen plaatsen. Hieronder een voorbeeld wat de inhoud van een v.e.d. zou kunnen zijn. Deze v.e.d. is voorzien voor 4 EPROMs, er is echter nog iets meer dan 1 EPROM leeg (ongeveer 68k).

Een kleine toelichting bij het schema. De ICs 1,2 en 9 zijn de buffers voor de adres- en datalijnen. De poorten IC10C,10B,10F en IC11 dekodieren het adres FFXD (enkel actief bij het schrijven) voor de latch IC7, die de hoogste adreslijnen A12'-A19' aanmaakt. De geheugenruimte DXXX, waarin de virtual disk actief is, wordt gedekodeerd door de poorten IC10A en IC13B. De v.e.d. wordt door poort IC12A in het hoogste blok van 256k geadresseerd. Een beveiliging tegen het schrijven naar de eproms, wordt gerealiseerd met de poorten IC10E en IC12B. Hieronder de kleine software-aanpassing van DOS65,

SDIR 3:

28-May-89 21:42 Disk: Virtual Eprom

```
>D          ECHO          PAUSE
EDDHLP.EDD  PLIST         TIME
PRINT       TYPE         cvar.mac
UNEXPAND    delC         BOOTLINK
direct      BOOTved      EXPAND
CAT         FORMAT       SDIR
HELP       SEROFF       cam
SERON      cc           mcom
cc.bin     smacro       DDOCTOR
DDOCTOR.DFT MEMFILL     SETDRIVES
SETMODE    cmat.mac    DPTIME
DUMP       OLOAD       SETRTC
```

71 files, 268 blocks free.

die noodzakelijk is om de v.e.d. door DOS65 te laten herkennen. Er wordt gebruik gemaakt van bepaalde zaken die reeds in DOS65 voorzien waren voor een winchester disk.

In principe is de v.e.d. enkel te gebruiken als er een virtual disk in het systeem zit. De RAM in het gebied D000..DFFF moet namelijk uitgeschakeld worden op het moment dat de v.e.d. wordt geadresseerd. Door middel van een eenvoudige hardware modificatie, kan de v.e.d. toch werken zonder virtual disk in het systeem. Deze wijziging houdt wel in dat er een verbinding loopt tussen de v.e.d. en de statische RAM-kaart. Van zodra er iets uit de v.e.d. ge-lezen wordt, gaat pin 6 van IC13A naar laag. Dit signaal wordt doorverbonden met pin 11 van N9 op de statische RAM-kaart, waardoor deze volledig geblokkeerd wordt. Op de statische RAM-kaart moet natuurlijk de draadbrug P verwijderd worden. Ter voorkoming van busconflikten, als de verbinding tussen de twee kaarten ontbreekt, wordt pin 11 van N9 op de RAMkaart best met een weerstand van 2k2 ohm met de massa verbonden. Ontbreekt de verbinding, dan werkt heel het systeem niet, maar beter dit dan enkele gesneuvelde databusbuffers. Voor de dynamische RAM-kaart bestaat er een ongeveer gelijkaardige oplossing. Het aanmaken van de EPROMs voor de v.e.d. kan eigenlijk het eenvoudigst als men beschikt over een virtual disk en een epromprogrammer. De virtual disk wordt geladen met alle programma's, die men in de v.e.d. wenst. Het verkegen bitpatroon van de virtual disk wordt in binary file's geladen, waarmee de EPROMs voor de v.e.d. geprogrammeerd kunnen worden. Indien men zelf geen EPROMprogrammer heeft, kan men met behulp van de volgende commandfile mbin &1 de binary files aanmaken. De diskette met de binary files kan je dan

```
SETUP       csys.mac    APPEND
cuser.mac   AS          EDITOR
ASN         EDTAB       RENAME
EP          RS232       VFORMATX
RS232.DAT   asdc        instalC
asmc        lmacro      CLEAR
login.com   COPY        LABEL
CREATE      LIST        SETBEGIN
MAP         SETCURS      ccas
cmac.mac    DELETE      MEMMOVE
MONITOR     SETPROMPT    cstr.mac
csub.mac    sprint
```

Fig. 1: Directory van EPROM- (virtual) disk

aan iemand bezorgen, die de EPROMs voor je wil programmeren.

```
; filename      :      mbin &1
; author       :      De Cock Luc
; date        :      6 january 1989
CL
ECHO This commandfile makes and saves the
      binary files needed
ECHO to program the eproms for the virtual
      eprom disk.
ECHO
ECHO Warning! Files in the virtual disk will be
      erased.
ECHO The virtual disk will be wiped clean (filled
      with $FF) and formatted.
ECHO
ECHO Everytime there is waited on your keystroke
      you can leave
ECHO this commandfile with Ctrl-C (all made
      binary files will be complete).
ECHO PAUSE Strike any key to continue
;FILLFF
VFORMAT
ECHO
nmtrck&1
copysort
mcbin1
PAUSE Make .bin for eprom 2? Yes : Strike a key
mcbin2
PAUSE Make .bin for eprom 3? Yes : Strike a key
mcbin3
PAUSE Make .bin for eprom 4? Yes : Strike a key
mcbin4
ECHO This completes mbin.
```

Zorg ervoor dat de programma's mbin, mcbin1-4, nmtrck1-4, restvirt, NAME en PAUSE op je systeemdisk staan. Het argument &1 van mbin is een getal ($1 \leq m \leq 4$) dat aangeeft hoeveel EPROMs er op de v.e.d-kaart zullen komen te staan. Dit argument bepaalt welke nmtrck* commandfile uitgevoerd zal worden en aldus hoeveel tracks en vrije blocks de virtual disk nog zal bevatten na wijziging van zijn bitmap. PAUSE is een variatie op ECHO met een ingebouwde lus, die wacht op een toetsaanslag. Het programma FILLFF is voor de werking van de v.e.d. niet nodig. Enkel wanneer de v.e.d. bekeken wordt met DDOCTOR, geeft het een netter uitzicht (in de lege sectors staat overal \$FF i.p.v. willekeurige data). Alle gewenste programma's worden door de commandfile copysort naar de virtual disk gecopieerd.

```
; file : nmtrk1
; function : places name, number of tracks and ad-
;           justs bitmap ;
;           of drive 2: for 1 EPROM 27512.
;select track 0 at 2XXX
MEMFILL FFF2,FFF2,01
;place name
LOAD NAME
;number of tracks : $10
MEMFILL 2021,2021,10
;adjust bitmap
MEMFILL 2070,20FF,00
;reselect original memory
MEMFILL FFF2,FFF2,0D

; file : nmtrk2
; for 2 EPROMs
MEMF FFF2,FFF2,01
LOAD NAME
LC 2021,2021,20
LC 2080,20FF,00
LC FFF2,FFF2,0D

; file nmtrk3
; for 3 EPROMs
MEMF FFF2,FFF2,01
LOAD NAME
LC 2021,2021,30
LC 2090,20FF,00
LC FFF2,FFF2,0D

; file : nmtrk4
; for 4 EPROMs
MEMF FFF2,FFF2,01
LOAD NAME
LC 2021,2021,40
LC 20A0,20FF,00
LC FFF2,FFF2,0D

; file : name.mac
; places v.e.d. name in the drive 2:
;
;           org      $2040
;
; name fcc 'Virtual Eprom' ;place name
;
;           end      name
een voorbeeld van de file copysort :
; file : copysort
COPY S:APPEND 2:
LC S:AS 2:
LC S:ASN 2:
LC S:CAT 2:
LC S:COPY 2:
LC S:CREATE 2:
LC S:DDOCTOR 2:
;enzovoort ...
```

De commandfiles mcbn* zorgen ervoor dat telkens 8 opeenvolgende tracks van de virtual disk worden geadresseerd in het geheugengebied 1000 8FFF, waarna ze op diskette worden gesaved. Vooraleer de commandfile te verlaten, wordt opnieuw het normale geheugen geselecteerd door de commandfile restvirt.

```
; file:      mcbn1
; function:   commandfile to make the the binary
              files for EPROM 1 of the v.e.d.
```

```
ECHO
ECHO EPROM 1
MEMFILL FFF1,FFF1,01 ;track 0 at 1000 1FFF
LC FFF2,FFF2,00      ;track 1 at 2XXX
LC FFF3,FFF3,1F      ;track 2 at 3XXX
LC FFF4,FFF4,1E
LC FFF5,FFF5,1D
LC FFF6,FFF6,1C
LC FFF7,FFF7,1B
LC FFF8,FFF8,1A
ECHO SAVE -b ep1a.bin 1000,8FFF
SA -b ep1a.bin 1000,8FFF
MEMFILL FFF1,FFF1,19
LC FFF2,FFF2,18
LC FFF3,FFF3,17
LC FFF4,FFF4,16
LC FFF5,FFF5,15
LC FFF6,FFF6,14
LC FFF7,FFF7,13
LC FFF8,FFF8,12
ECHO SAVE -b ep1b.bin 1000,8FFF
SA -b ep1b.bin 1000,8FFF
restvirt
```

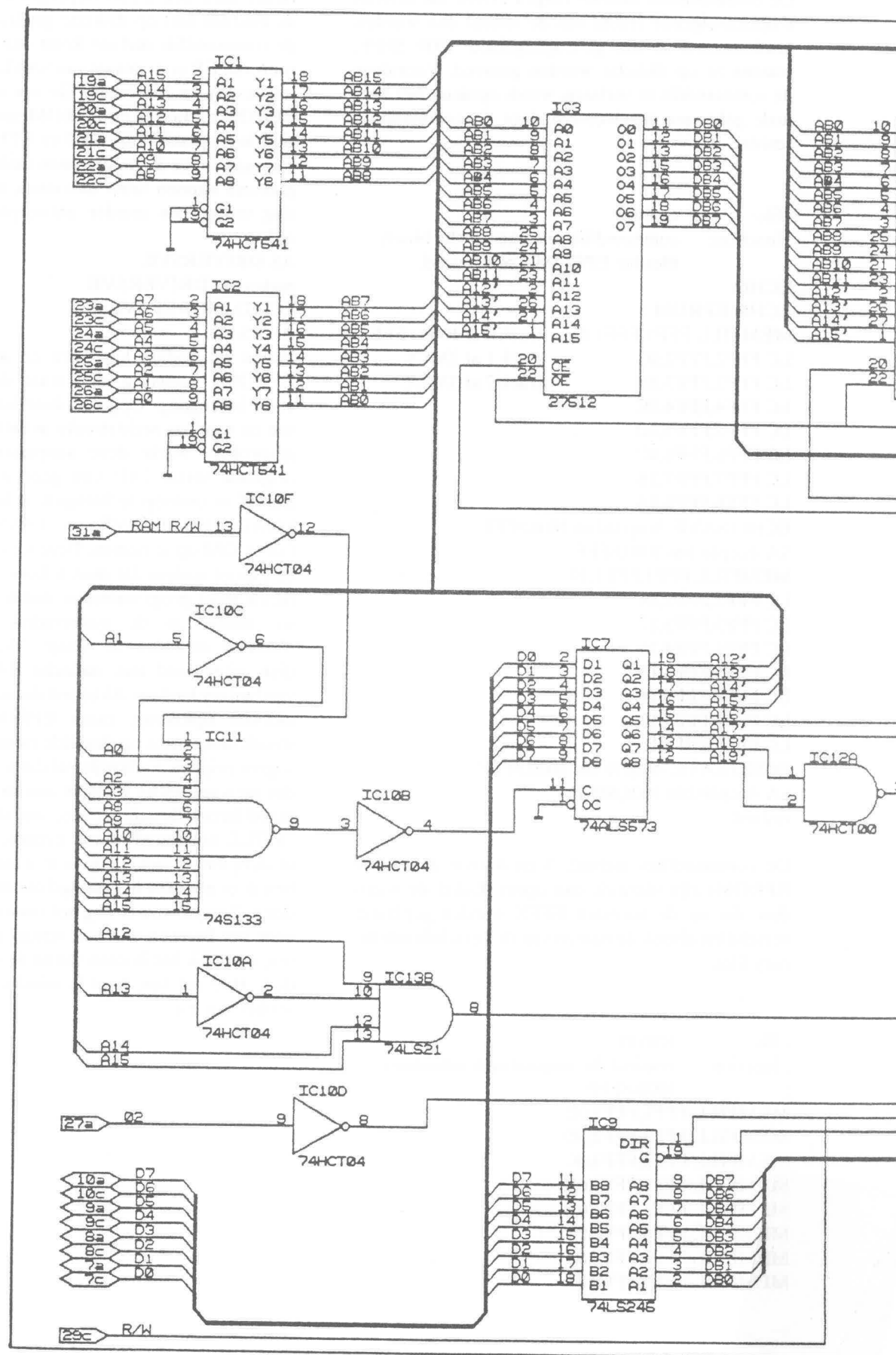
De commandfiles mcbn2, 3 en 4 voor de andere EPROMs zijn identiek dan opzet. Enkel de waarden, die op de adressen FFFX worden geplaatst, verschillen alsook de namen van de verschillende binary files.

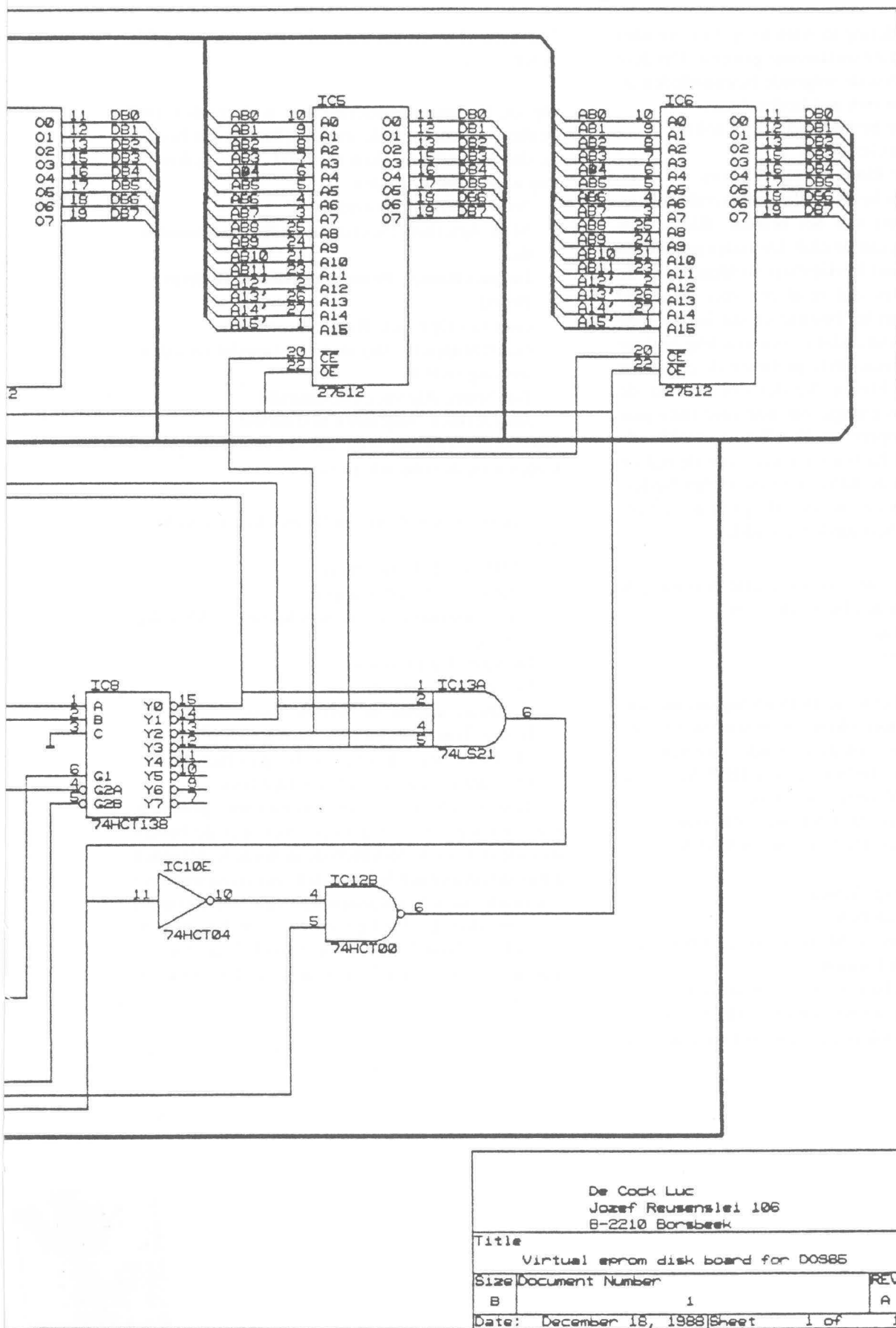
```
; file:      restvirt
; function:   reselect the original systemmemory
;            1000-8FFF
MEMFILL FFF1,FFF1,0E
MEMFILL FFF2,FFF2,0D
MEMFILL FFF3,FFF3,0C
MEMFILL FFF4,FFF4,0B
MEMFILL FFF5,FFF5,0A
MEMFILL FFF6,FFF6,09
MEMFILL FFF7,FFF7,08
MEMFILL FFF8,FFF8,07
```

Heeft men een EPROMprogrammer, dan hoeven de bin.files niet op diskette geplaatst te worden. In de commandfile mcbn* komt dan telkens een aanroep van EP in de plaats van SAVE Let op dat je telkens bij de 2de binary file van elke EPROM het EPROMstartadres en EPROMeindadres in EP wijzigt in respectievelijk 8000 en FFFF. Hetgeen deze command files doen, zou men makelijk door 1 programma kunnen laten uitvoeren, maar deze oplossing vroeg toen minder ontwerptijd en het werkt ook.

```
AS DRIVERSVE
makecom DRIVERSVE
LOAD DRIVERSVE
ASN S=3:
```

Indien de nodige hardware en geprogrammeerde EPROMs aanwezig zijn, staat de epromdisk ter uwer beschikking. Opgelet: doordat bepaalde adressen en routines rechtstreeks in DOS65 worden aangesproken, werkt deze aanpassing enkel bij de originele versie 2.01! Om geen nieuwe versie van BOOT in omloop te brengen, is het aan te bevelen LOAD DRIVERSVE en 3:ASN S=3: in LOGIN.COM op te nemen. Deze v.e.d. heeft natuurlijk een groot nadeel dat men telkens de EPROMs opnieuw moet programmeren, wanneer men een nieuwe utiliteit in de systeemdisk wil bijplaatsen. Vandaar misschien de vraag: "Waarom geen ram disk, uitgevoerd met statische RAMs en voorzien van battery backup. Akkoord dat is waarschijnlijk de mooiste oplossing, maar EPROMs bieden nog steeds meer bytes op dezelfde ruimte tegen een gunstigere prijs en zodikwijls zal dat toch niet gebeuren, dat men iets moet wijzigen aan de systeemdisk. Iemand bezorgde mij het idee om als de foutmelding "3:FILE not found" komt, eventueel te gaan kijken of deze FILE niet op drive 0: staat. Maar voorlopig ben ik er nog niet in geslaagd om met deze tip iets te doen. Zowaar zou ik nog het voornaamste argument voor het bouwen van een virtual epromdisk vergeten, namelijk het booten vanuit de de virtual epromdisk. Hoe dat kan, leest U misschien in een volgend artikel van mij.





Taakverdeling bestuursleden.

Op de ledenvergadering in Almelo op 11 november 1989 is er een bestuursverkiezing geweest. Op deze ledenvergadering zijn de volgende bestuursleden afgetreden en stelden zich niet herkiesbaar:

- 1: Het algemeen bestuurslid Adri Hankel
- 2: De secretaris Gert Klein
- 3: De voorzitter Rinus Vleesch Dubois

Deze drie personen hebben allen gedurende vele jaren deel uitgemaakt van het bestuur, Rinus zelfs sinds de oprichting van de club. De redenen waarom de personen zich niet herkiesbaar stelden was in de eerste plaats het feit dat ze al een zeer lange tijd deel uitmaakten van het bestuur en dat het voor de club van groot belang is dat er een krachtig bestuur komt met frisse ideeën om te proberen de club weer nieuw leven in te blazen. Verder ontbrak het de mensen aan tijd en energie om nog voor twee jaar als bestuurslid te opereren. Ik wil de vertrekkende bestuursleden heel hartelijk danken voor de tijd en het werk dat ze in de KIM Gebruikersclub Nederland gestoken hebben en voor de prettige samenwerking die we als bestuursleden hadden.

Op de vergadering zijn voor deze drie personen de volgende mensen in de plaats gekomen:

- 1: Mick Agterberg
- 2: Geert Stappers
- 3: Ton Smits

Hiermee is wat wel eens de DOS-65 hegemonie binnen het bestuur doorbroken. De bestuursleden bezitten/gebruiken namelijk de volgende systemen:

- 1: Nico de Vries: DOS-65 en een IBM AT kloon
- 2: Ton Smits: EC-65(k) en Junior
- 3: Geert Stappers: Atari ST onder OS9/68k
- 4: Jacques Banser: DOS-65 en een IBM AT kloon
- 5: Mick Agterberg: Amiga
- 6: Jan Derksen: DOS-65
- 7: Gert van Opbroek: MC68000 onder OS9/68k, Junior en IBM XT-kloon

Kortom, niet alleen binnen de club maar ook binnen het bestuur hebben we nu een grote diversiteit aan systemen. Ik hoop ook in het clubblad voor al deze

(en andere) systemen geregeld iets te mogen publiceren.

Op de bestuursvergadering van 8 december 1989 hebben we, conform de statuten, binnen het bestuur de diverse bestuurstaken verdeeld. Deze taakverdeling is als volgt geworden:

Nico de Vries: Voorzitter

Mick Agterberg: Secretaris en ledenadministratie

Jacques Banser: Penningmeester en Bulletin Board

Gert van Opbroek: Redactiesecretaris

Geert Stappers: Algemeen bestuurslid en ledenwerving en P.R.

Ton Smits: Algemeen bestuurslid

Jan Derksen: Algemeen bestuurslid

Voor ondersteuning voor uw systeem kunt u aankloppen bij de volgende personen:

Algemene vragen: Gert van Opbroek en Geert Stappers

AMIGA: Mick Agterberg

ATARI ST: Geert Stappers

Datacommunicatie: Jacques Banser en Mick Agterberg

DOS-65: Jan Derksen

EC-65(k): Ton Smits

Hardware algemeen: Nico de Vries

Junior: Ton Smits en Gert van Opbroek

MS-DOS: Nico de Vries en Jacques Banser

Programmeertalen: Gert van Opbroek

Heeft u vragen op één van bovenstaande gebieden, dan kunt u gerust contact opnemen met de bovenstaande personen. Mochten de genoemde personen u niet direct kunnen helpen, dan zoeken zij wel weer een derde die u behulpzaam kan zijn. Als u contact opneemt, doe dat dan bij voorkeur schriftelijk of via het bulletin board. Wilt u ons toch bellen, doe dat dan in ieder geval na 7:30 en liever ook niet tijdens het journaal van 8 uur.

Gert van Opbroek

Van de voorzitter.

Vrijdag 8 december was er weer de gebruikelijke bestuursvergadering. Toch was er iets bijzonders aan deze vergadering: er was geen voorzitter, en ook geen secretaris. Bij de laatste ledenvergadering traden Rinus Vleesch Dubois en Gert Klein als respectievelijk voorzitter en secretaris af.

De vereniging is beide ex-bestuursleden veel dank verschuldigd voor het vele werk dat zij voor de vereniging hebben verzet. Het bedankje in natura dat ze mee naar huis kregen was daar maar een magere afspiegeling van.

Toch staat er boven dit stukje: 'Van de voorzitter'. Want de KIM gebruikersclub Nederland is inmiddels weer een secretaris en een voorzitter rijker. Dat was een van de uitloeselen van diezelfde vergadering op 8 december. Omdat misschien niet iedereen even duidelijk zal zijn hoe bestuursleden aan hun functies komen, het volgende.

De leden van onze vereniging kiezen uit de naar voren getreden kandidaten voor een bestuursfunctie een bestuur, dat minimaal uit drie leden moet bestaan, zo vermelden de statuten. Het huidige bestuur telt zeven leden: u vindt hun namen en adressen voorin dit blad.

De statuten bepalen verder, dat het bestuur zelf de onderlinge taakverdeling moet regelen. Dus was op deze bestuurvergadering het belangrijkste agenda-punt: de taakverdeling binnen het bestuur. Binnen het bestuur bleken er twee kandidaten voor de taak van secretaris te zijn, te weten Mick Agterberg en Ton Smits. Drie bestuursleden bleken bereid de functie van voorzitter te vervullen: Mick Agterberg, Jan Derksen en Nico de Vries. Zoals ook door de statuten wordt bepaald, werd er schriftelijk gestemd over wie nu wat ging doen. Mick Agterberg werd na 1 stemronde tot secretaris verheven. Ook voor de voorzittersfunctie was slechts 1 stemronde voldoende. Wie de voorzitter werd, ziet u hieronder.

Het kolommetje is al weer bijna vol. Rest mij nog u te beloven dat ik mijn uiterste best zal doen in mijn nieuwe nog onwennige functie. En natuurlijk wens ik een ieder opperbeste feestdagen toe en dat het nieuwe jaar brengen moge wat u ervan verwacht.

Tot ziens in Krommenie.

Nico de Vries



Offset drukkerij
Blekerstraat 72
7513 DX ENSCHEDE
Tel.: 053 - 31 05 72

COMPLEET IN DRUKWERK

- in oplages van 1 tot duizend in alle formaten tot 50 x 70 cm, in één of meer kleuren.
- specialist in compleet drukwerk (zetten, drukken, vergaren, afwerken) zoals idem sets, catalogi, readers, dissertaties, periodieken.
- korte levertijden door combinatie van drukken, afwerken, postverwerking alles onder één dak.
- Ultra korte leveringstijden voor periodieken, strooifolders mogelijk. Bijv. 300 boeken van 160 pagina's in twee dagen!

NIEUW

- A2 posters
- twee kleuren drukpers voor briefpapier, folders en enveloppen in meer kleuren

Dikhuidige Personeelsindeling

Een splinternieuwe methode voor beroepskeuze bij high-tech personeel!

De afgelopen jaren zijn duizenden manjaren en miljoenen guldens gespendeerd aan het vinden van de juiste man voor de juiste baan. Dit geldt zeker voor technisch hoogstaande bedrijven waar talent een schaars, dus duur, goed is. Onlangs echter, hebben jaren van intensieve studie door de meest uitgelezen deskundigen op het gebied van psychoindustriële interpersonele optimalisatie geresulteerd in de ontwikkeling van een eenvoudige doch trefzekere test om de beste koppeling te vinden tussen persoon en professie. Nu is het eindelijk mogelijk om onfeilbaar de juiste baan te selecteren.

De werkwijze is eenvoudig: Eenieder wordt naar naar Afrika gezonden om op olifanten te jagen. Het daaruit resulterende gedrag bij de jacht op olifanten wordt dan gecategoriseerd door toetsing aan de hieronder besproken regels, waarbij de beste benadering van het waargenomen gedrag de classificatie bepaalt.

Regels voor classificatie

Wiskundigen jagen op olifanten door naar Afrika te gaan, alles te verwijderen wat geen olifant is, en een exemplaar te vangen van wat overblijft. Ervaren wiskundigen zullen trachten het bestaan van tenminste een unieke olifant te bewijzen, alvorens bij stap 1 verder te gaan. Professoren in de wiskunde bewijzen het bestaan van tenminste een unieke olifant, en laten vervolgens het daadwerkelijk opzoeken en vangen van een olifant aan hun studenten over.

Computerdeskundigen jagen op olifanten door het uitvoeren van Algoritme A:

1. Ga naar Afrika.
2. Start bij Kaap de Goede Hoop.
3. Werk noordwaarts op geordende wijze, waarbij het continent afwisselend oost- dan wel westwaarts doorkruist wordt.
4. Tijdens elke doorkruising:
 - a. Vang elk dier dat wordt waargenomen.
 - b. Vergelijk dit met een dier waarvan zeker is dat het een olifant is.
 - c. Stop als de vergelijking waar is.

Ervaren computerprogrammeurs breiden Algoritme A uit door een olifant in Caïro te plaatsen, om zeker te zijn van een eindig algoritme. Machinetaalpro-

grammeurs voeren Algoritme A bij voorkeur op handen en knieën uit.

Ingenieurs jagen op olifanten door naar Afrika te gaan, willekeurig grijze dieren te vangen, en te stoppen als er een binnen een marge van 15%, het gewicht van een tevoren geobserveerde olifant benadert.

Economisch deskundigen jagen niet op olifanten, maar geloven dat olifanten op zichzelf gaan jagen als je ze maar genoeg betaalt.

Statistici jagen op het eerste dier dat ze één keer waarnemen en noemen dat een olifant.

Economisch deskundigen jagen niet op olifanten, maar geloven dat olifanten op zichzelf gaan jagen als je ze maar genoeg betaalt.

Consultants jagen niet op olifanten, en hebben waarschijnlijk nog nooit ergens op gejaagd, maar mensen die wel jagen kunnen ze per uur huren voor advies. Consultants kunnen tevens de correlatie bepalen tussen hoedmaat en kogelkleur en de efficiëntie van een zekere jachtstrategie, als iemand anders maar een olifant identificeert.

Politici jagen niet op olifanten, maar ze zullen de door anderen gevangen olifanten delen met hun kiezers.

Advocaten jagen niet op olifanten, maar ze volgen de kudde, discussiërend over het feit wie de eigenaar van de uitwerpselen is. Software-advocaten zullen de hele kudde opeisen, gebaseerd op de 'look and feel' van een uitwerpsel.

Vicepresidenten van research- en ontwikkelafdelingen doen hun uiterste best om op olifanten te jagen, maar hun staf is ontworpen om dat te voorkomen. Als de vicepresident er toch toe komt om op olifanten te gaan jagen, zal zijn staf trachten te verzorgen dat alle voorkomende olifanten al zijn weggejaagd, voordat de vicepresident de kans krijgt ze te zien. Mocht de vicepresident een niet weggejaagde olifant waarnemen, zal zijn staf (1) de vicepresident complimenteren met zijn goede waarnemingsvermogen en (2) zich dusdanig uitbreiden dat herhaling uitgesloten is.

Senior Managers stippelen een brede olifantenjacht-politiek uit, gebaseerd op de aanname dat olifanten

net grote veldmuizen zijn, met alleen een wat lagere stem.

Inspecteurs belast met kwaliteitscontrole negeren de olifanten en zoeken naar fouten die de andere jagers maakten bij het inpakken van de jeep.

Verkopers jagen niet op olifanten, maar komen hun tijd door met het verkopen van olifanten die ze niet gevangen hebben, en beloven die twee dagen voor de opening van het seizoen te leveren. Software-verkopers sturen het eerste wat ze vangen op, en schrijven ongeacht de vangst een rekening voor een olifant. Hardware-verkopers vangen konijnen, schilderen ze grijs en verkopen ze als desktop-olifanten.

Geldigheidscontrole

Tijdens een onderzoek naar de geldigheid van deze regels kwam het volgende naar voren:

Alle mensen die aan het onderzoek deelnamen waren geldig. De weinigen die dat niet waren, verwachtten daar spoedig van te herstellen. Uit dit onderzoek werd een statistisch vertrouwensniveau afgeleid. Vijfennegentig procent van de onderzochte personen bleek tenminste 67 procent vertrouwen in statistieken te hebben.

D. Zwietering

Bron: New--NewTon december '89

Met dank aan Stop Bit, BYTE september 1989



Computers (Deel 5)

Inleiding

In het juni-nummer had ik aangegeven dat het vervolg op deel 4 in het augustusnummer geplaatst zou worden. Dat is dus niet helemaal gelukt. Het augustusnummer is om diverse redenen niet verschenen en bij het maken van het oktobernummer ontbrak mij de tijd en de inspiratie om nog iets aan deze serie te doen. Dat zullen we dus nu maar even rechtzetten.

In de vorige aflevering hebben we het gehad over het zogenaamde programmeermodel van de processor. Zoals in deel vier uitgelegd is, geeft het programmeermodel aan hoe de processor intern opgebouwd is, het geeft aan welke registers de processor heeft, hoe groot die registers zijn etc.

Verder hebben we het gehad over de stack, wat daarmee wordt bedoeld en hoe de stack gebruikt wordt en tenslotte hebben we van drie typen processoren de adresseermethoden behandeld.

In deze aflevering zullen we eens naar de instructies gaan kijken die een microprocessor aan boord heeft. Hierbij moet ik helaas de 8088 buiten beschouwing laten om de simpele reden dat ik daar totaal niets van af weet. Ik heb begrepen dat Nico in zijn serie over MS-DOS hier nog iets aan wil gaan doen.

Lengte van instructies

Zoals in deel 3 aangegeven is, kan een microprocessor alleen iets doen als hij een opdracht krijgt om iets te doen. Hij haalt deze opdracht uit het geheugen en wel van het adres dat op dat moment in de program counter staat. Verder is daar verteld dat een instructie niet altijd gelijke lang hoeft te zijn. Een 6502 kent bijvoorbeeld instructies van 1, 2 of 3 byte die ook in resp. 1, 2 of 3 slagen opgehaald worden. De 68000 is wat dat betreft iets minder zuinig met geheugen, deze processor kent instructies van 2, 4, 6, 8 en 10 byte die, vanwege de 16 bits brede databus ook in 1, 2, 3, 4 of 5 slagen opgehaald worden.

De lengte van een instructie wordt over het algemeen bepaald door de adresseermethode. Zo werkt op een 6502 een instructie met een lengte van 1 byte altijd op de processor zelf. Een instructie met een lengte van 2 byte werkt bij de 6502 op pagina 0, dus de eerste 256 bytes van het geheugen of het is een instructie met immediate adressering. Een instructie met een lengte van 3 byte werkt ook op het geheugen, alleen staat er dan een compleet adres van 16 bits in de instructie. Een drie byte instructie wordt op de 6502 dus gebruikt bij absolute adressering.

Blijven we nog even bij de 6502, dan kunnen we nog wat meer leuke dingen over de instructie-coderingen vertellen. Zoals ook al in deel 3 verteld is, bevat het eerste byte een code die aangeeft welke instructie en in welke adresseermethode bedoeld wordt. Dit eerste byte heet daarom ook wel "Opcode". Een 6502 zou met zijn opcode van 8 bits dus maximaal 256 verschillende instructies kunnen hebben. De tweede byte bevat bij de 6502 altijd het minst significante byte van het adres dat benaderd moet worden. Verder bevat het eventuele derde byte het meest significante byte van het adres. Waarschijnlijk is het feit dat alleen op deze manier zowel bij page zero en absolute adressering het tweede byte altijd het minst significante deel van het adres bevat de reden dat op de 6502 het minst significante byte op het laagste adres staat. Dit is dus op de 6502 consequent doorgevoerd.

Bij de 68000 hebben we dus te maken met instructies van 2, 4, 6, 8 en 10 byte. Hierbij werkt een instructie van 2 byte meestal op de processor zelf. Er worden bijvoorbeeld gegevens van het ene register in een andere gekopieerd. Verder past een sprongopdracht waarbij het doel minder dan 128 byte vooruit of terug ligt ook in een twee-byte instructie. Een derde mogelijkheid is dat er in de instructie een kleine constante staat die bijvoorbeeld in een register overgenomen moet worden. Een instructie met een lengte van vier byte kan een constante van 16 bits bevatten. Dit kan bijvoorbeeld een verplaatsing zijn of een constante die met behulp van immediate adressering verwerkt moet worden. Verder kent de 68000 ook een gebied dat, net als bij de 6502 met pagina 0, met een incompleet adres benaderd kan worden. Dit is het geheugen gebied tussen \$00000000 en \$0000FFFF, dus de eerste 64 kb. Ook in dit geval wordt alleen het minst significante deel van het adres gebruikt. Een instructie met een lengte van 6 byte kan een compleet adres van 4 byte bevatten of een constante van 32 bit. Bovendien zou ze twee adressen van 16 bit kunnen bevatten of een 16 bit constante en een twee byte adres. Een instructie met een lengte van 8 byte bevat een combinatie van een 16 bits constante en een 32 bits adres of andersom of een 16 bits adres gecombineerd met een 32 bits adres. Een instructie met een lengte van tien byte bevat twee adressen van 32 bits of een constante van 32 bits en een volledig adres.

Instructies van een 6502

In het volgende overzicht zijn de instructies van de 6502 aangegeven. Deze instructies kunnen nog gecombineerd worden met een aantal adresseermethoden waarbij de toegestane mogelijkheden wel afhankelijk zijn van de instructie.

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with accumulator
ASL	Shift Left One Byte (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result Not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-OR" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Flag
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory

TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Register
TYA	Transfer Index Y to Accumulator

Totaal heeft de 6502 56 instructies.

Behalve de 6502 bestaan er ook minstens twee versies van een C-MOS uitvoering van de 6502 die met 65C02 aangegeven wordt. Dit type processor heeft een aantal instructies meer als de gewone 6502. Bovendien hebben een aantal instructies meer adresseermogelijkheden. De 65C02 kan alle programma's draaien die ook op de 6502 lopen mits er in het programma geen gebruik is gemaakt van niet gespecificeerde instructies. De 6502 gebruikt namelijk niet alle 256 waarden die de opcode kan krijgen. Nu wil het geval dat voor een aantal van de niet gebruikte waarden de processor wel iets doet en vaak ook nog hele nuttige dingen. Natuurlijk is het niet de bedoeling dat dergelijke instructies gebruikt worden maar er bestaan programma's die toch gebruik maken van dergelijke instructies. Op een 65C02 bestaan deze illegale instructies niet en dus zullen deze programma's niet op een 65C02 draaien.

Het lijkt me niet zinvol in het kader van deze serie alle instructies van de 6502 of welke andere processor dan ook te bespreken. Er bestaan enkele zeer goede boeken voor dit doel en voor een diepgaande behandeling van het instructieset verwijs ik daar dan ook naar. Wel heeft het zin iets meer structuur in het instructieset aan te brengen. In het bovenstaande overzicht is het instructieset alfabetisch weergegeven. Hier zit dus eigenlijk geen enkele structuur in. Een andere aanpak zou kunnen zijn, de instructies op basis van wat ze doen bij elkaar te nemen. We kunnen dan bij de 6502 de volgende groepen onderscheiden:

1: Verplaatsing van gegevens.

In deze groep vallen LDA, LDX en LDY en hun partners de STO, STX, en STY -instructies om informatie tussen de registers en het geheugen uit te wisselen. Verder alle transfer-instructies dus TAX, TAY, TSX, TXA, TYA, TXS. Ook de instructies die gegevens op de stack schrijven (push) of gegevens van de stack halen (pull) vallen in deze categorie. Dit zijn PHA, PHP, PLA en PLP.

2:Arithmetische instructies.

Deze categorie bevat alle instructies die een rekenkundige bewerking uitvoeren. Dit zijn niet alleen de instructies voor optellen en aftrekken (ADC en SBC) maar ook de instructies voor het verhogen en verlagen van de inhoud van een register of een ge-

heugen plaats dus: INC, DEC, INX, DEX, INY, DEY. Ook de vergelijk-instructies CMP, CPX en CPY horen in deze groep thuis.

3: Logische instructies.

Deze instructies voeren logische bewerkingen uit. De categorie bestaat dus uit de instructies AND, EOR, OR, verder is de BIT-instructie waarmee afzonderlijke bits bekeken kunnen worden onderdeel van deze groep.

4: Instructies voor schuiven en roteren.

Deze instructies verplaatsen de bits van de inhoud van een register of geheugenplaats één positie naar links of rechts. Bij een roteer-operatie wordt de carry in het byte gestopt en wordt het bit dat uit het byte komt in de carry gestopt; bij een schuifoperatie wordt er een 0 in het byte geschoven en komt het bit dat uit het byte komt in de carry. In deze categorie kent de 6502 de volgende instructies: ASL, LSR, ROL en ROR.

5: Programma besturing.

Deze instructies bepalen de loop van een programma. Dit zijn dus alle spronginstructies BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, JMP en JSR. Verder uiteraard de RTS en de NOP.

6: Systeembesturing.

Met deze instructies kan de processor zelf bestuurd worden. De 6502 kent in deze groep instructies voor het wijzigen van de inhoud van het status-register, de CLC, CLD, CLI en CLV en de SEC, SED, SEI-instructie. Verder behoren de BRK en de RTI die te maken hebben met interrupts ook in deze groep thuis.

Instructies van de 68000

In het onderstaande overzicht is het instructieset van de 68000 weergegeven. Ook in dit geval kunnen de instructies, afhankelijk van de instructie, in combinatie met diverse adresseermethoden gebruikt worden. Verder kennen diverse instructies nog zogenaamde varianten, dit zijn speciale uitvoeringen van de instructie.

ADBC	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
Bcc	Branch Conditionally (1)
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always

BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DBcc	Test Condition, Decrement and Branch (2)
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive OR
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical OR
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditional (3)
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Notes:

Definitie van de Condition Codes:

- 1) cc = CC: Carry Clear
- CS: Carry Set
- EQ: Result is Zero (Equal)
- GE: Greater or Equal (2's complement)
- GT: Greater Than
- HI: Higher (Unsigned)
- LE: Less or Equal (2's complement)
- LS: Lower or Same (Unsigned)
- LT: Less Than
- MI: Minus (0)

NE: Not Equal (Result is unequal Zero)

PL: Plus (=0)

VC: No Overflow (V Clear)

VS: Overflow (V Set)

2) cc = Zie note 1 +

F: False

T: True

DBcc vormt een soort REPEAT/UNTIL Loop. Dat wil zeggen dat in het programma een loop uitgevoerd wordt totdat aan de aangegeven conditie voldaan is. Bovendien wordt er een data-register aangegeven waarvan de inhoud iedere keer met één verlaagd wordt. Als het laagste woord in dit data register gelijk wordt aan -1, dan wordt de loop ook beëindigd.

3) cc = Zie note 2

Met behulp van de Scc-instructie kan een conditie overgenomen worden in een byte. Als aan de conditie voldaan is, dan wordt de inhoud van dit byte \$FF, is er niet aan voldaan, dan wordt de inhoud \$00.

Behalve de bovengenoemde instructies kent de 68000 nog enkele varianten van de genoemde instructies. Dit zijn instructies die strikt genomen een afzonderlijke instructie zijn maar zeer veel lijken op één van de genoemde instructies. Voorbeelden hiervan zijn de varianten van de MOVE-instructie:

MOVE Verplaats gegevens

MOVEA Verplaats een adres

MOVEM Verplaats de inhoud van meerdere registers, met name bedoeld voor bouwstenen met een 8 bits brede databus die aangesloten worden op de 16 bits brede bus van de 68000

MOVEP Verplaats gegevens van perefere-bouwstenen

MOVEQ Verplaats een kleine (8 bit) constante

MOVE from SR Verplaats uit het status register

MOVE to SR Verplaats naar het status register

MOVE to CCR Verplaats naar het user byte

MOVE USP Verplaats de user stack pointer

Ook bij een aantal andere instructies komen dergelijke varianten voor. Mensen die in assembler programmeren op een 68000-systeem zullen waarschijnlijk zeer goed begrijpen waarom de 68000 een Complexe Instruction Set Computer (CISC) genoemd wordt, de 68000 kent zeer veel verschillende instructies met een groot aantal adresseermethoden. Wat ik zelf heel vervelend hieraan vind is het feit dat niet alle instructies dezelfde adresseermethoden hebben. Zou dit wel zo zijn, dan zouden we te maken hebben met een zogenaamde "Orthogonale" processor. Helaas is dit niet zo en dat betekent dat het me vrij geregeld overkomt dat de processor net niet die combinatie van instructie en adressering heeft die ik graag zou willen gebruiken maar helaas, het is niet anders.

De instructies van de 68000 kunnen natuurlijk ook weer in groepen onderverdeeld worden:

1: Verplaatsing van gegevens

In deze categorie vallen EXG, LEA, LINK, MOVEM, MOVEP, MOVEQ, PEA, SWAP en UNLK. Hiervan zijn LINK en UNLK wat vreemde instructies. De LINK-instructie reserveert een hoeveelheid ruimte op de stack, de UNLK (Unlink) -instructie geeft deze ruimte weer vrij. Deze instructies worden met name door de vertalers van hogere programmeertalen gebruikt om ruimte te maken voor variabelen.

2: Aritmetische instructies.

In deze groep heeft de 68000 de volgende instructies: ADD (met als varianten ADDA, ADDI, ADDQ en ADDX), CLR, CMP (met als varianten CMPA, CMPI en CPM), DIVS, DIVU, EXT, MULS, MULU, NEG (en de variant NEGX) SUB (met als varianten SUBA, SUBI, SUBQ en SUBX), TAS en TST. Binnen de hier genoemde varianten staat A voor adres, I voor immediate, Q voor Quick dus een kleine constante, M voor Memory en X voor eXtended. Vooral het laatste is een kleine uitleg waard. De 6502 gebruikt bij optellen en aftrekken altijd de carry. Bij de 68000 wordt de carry alleen gebruikt voor het resultaat. Bij instructies met X aan het eind wordt de X-vlag gebruikt om de carry van de vorige bewerking te verwerken. Bij ADDX worden dus niet de beide operanden bij elkaar opgeteld maar wordt ook de inhoud van de X-vlag bij het resultaat opgeteld.

3: Logische instructies.

Hiervan heeft de 68000 de volgende: AND (en ANDI), OR (en ORI), EOR (en EORI) en de NOT.

4: Schuiven en roteren.

Hiervan vinden we bij de 68000 de volgende instructies: ASL, ASR, LSL, LSR, ROL, ROR, ROXL en ROXR. De ROXL en ROXR gebruiken tijdens het roteren weer de X-vlag. In tegenstelling tot de 6502 kan men bij de 68000 opgeven hoeveel bits er in de instructie geschoven of geroteerd dienen te worden.

5: Bitmanipulaties.

Dit zijn instructies waarmee losse bits gewijzigd kunnen worden. Aanwezig zijn BTST, BSET, BCLR en BCHG.

6: BCD-instructies.

In het programmeermodel van de 6502 hebben we gezien dat de 6502 middels een vlag in de toestand BCD (Binary Coded Decimal) gezet kan worden.

De 68000 kent een aantal speciale instructies voor BCD-berekeningen. Dit zijn: ABCD, SBCD en NBCD.

7: Instructies voor programmabesturing.

Deze groep bevat de instructies: Bcc, DBcc, Scc, BSR, JSR, RTS, JMP en RTR. Hierin staat de afkorting "cc" voor Condition Code.

8: Systeem besturing.

In deze groep heeft de 68000 de instructies: MOVE USP waarmee de processor in de zogenaamde Supervisor Mode de user stack kan benaderen, RESET, RTE, STOP, CHK, TRAPV en TRAP. De CHK, TRAPV en TRAP-instructies genereren altijd of afhankelijk van bepaalde omstandigheden (bijvoorbeeld er is Oververflow opgetreden) een zogenaamde Exception. Dit is te vergelijken met een interrupt waarbij de processor dus een ander stuk programma (service-routine) uit gaat voeren.

Afsluiting.

Ik heb in deze aflevering van de serie getracht een overzicht te geven van de instructies die zoal in een microprocessor aanwezig kunnen zijn. Het is niet

mijn bedoeling om elke instructie tot op de bodem uit te diepen, daar zijn boeken voor, bijvoorbeeld degene die in de literatuurlijst vermeld staan. Helaas kan ik niets over het instructieset van de 8088 en aanverwante processoren vertellen, ik ben er echter van overtuigd dat Nico de Vries in zijn serie hier wel op in zal gaan.

Literatuur.

Voor mensen die meer over de 6502 willen weten, kan ik bijvoorbeeld de volgende boeken aanraden:

1: A. Nachtmann en G.H. Nachbar: Junior Computer 1 t/m 4

2: R. Zaks: Programming the 6502 (volgens mij ook in het nederlands verkrijgbaar).

Voor de 68000 zijn er ook diverse boeken, het boek dat ik gebruik is:

3: S. Williams: Programming the 68000.

Gert van Opbroek

Errata.

In de vorige delen van de serie "IBM-PC en z'n klonen" zijn helaas wat kleine foutjes geslopen. Hier volgen de verbeteringen:

Deel 2, paragraaf 6:

MOV JOOP,AX schrijft natuurlijk de inhoud van AX in de geheugenplaats JOOP (16-bit).

Deel 2, paragraaf 8:

Helaas emuleert de V30 ook alleen de Intel 8080. Dus geen Z80.

Deel 4, paragraaf 4:

Op I/O adres 1F0-1F8 zit in een XT niet de harddisk controller: dit is het adres in een AT. In een XT zit de controller op de adressen 320-327.

Verder is het door de constructie van het moederbord zo, dat het I/O-gebied van 100 t/m 1FF niet gebruikt kan worden. Deze beperking geldt niet voor de AT en een aantal XT-klonen.

Deel 5, paragraaf 6:

De CRTC op de EGA kaart bevindt zich altijd op adres 03X4 en 03X5, waarbij X D is bij alle kleur modes, en B bij alle monochrome modes. Op adres 03C4 en 03C5 zitten op de EGA kaart speciale hulpregisters, die het gedrag van de kaart nader bepalen. Zo emuleert de EGA in de CGA grafische modes het merkwaardige even/oneven verschijnsel volledig. Dit wordt met deze registers ingesteld.

Een verhaal betreffende bulletin borden.

In dit artikel wil ik in het kort vertellen waar een Bulletin Board System in eerste instantie voor bedoeld is, namelijk het uitwisselen van berichten. Het idee ontstond pakweg een 10 jaar geleden toen enkele personen tussen elkaar berichten wilden gaan uitwisselen. Dezen hebben toen een klein pakketje geschreven wat hen in staat stelde elkaar berichten te sturen. In de loop van de tijd zijn die pakketten zodanig uitgegroeid tot datgene wat men tegenwoordig kent.

Tegenwoordig is het namelijk niet alleen voor de bezitter van het/een Bulletin Board mogelijk om berichten te lezen en te versturen, maar ook anderen die van buiten komen (de zg. users) kunnen berichten lezen en ingeven. Verder zijn de huidige generatie BBS-programma's zo uitgebreid dat het bijna complete graphische pakketten zijn geworden. Ook kunnen er verschillende externe utilities aan het BBS gehangen worden die het dan weer mogelijk maken dat de gebruikersvriendelijkheid toeneemt of alleen de mogelijkheden van het BBS uitgebreid worden.

Zoiets als het versturen van files (dus het bekende up- en downloaden) was er in het begin helemaal niet bij. Die optie is er later bijgekomen als leuke service voor de gebruiker. Helaas is het tegenwoordig zo dat men alleen maar naar een Bulletin Board System belt om b.v. een programmaatje te downloaden. Een bekende kreet is b.v. het 'sponzen'. Wat dus wil zeggen dat men zéééér veel download (haalt) zonder ook maar iets voor anderen achter te laten.

Was het in het verleden bijvoorbeeld alleen maar mogelijk om via vijf verschillende protocollen een file te versturen; tegenwoordig zijn dat er rond de twintig. Vooral ZModem is populair. Zo is dit protocol het snelst dat op een BBS kan draaien, heeft een zeer goede errorafhandeling en er bestaat zelfs een mogelijkheid om een download die de eerste keer maar half is voltooid (bv. onderbreking wegens een slechte telefoonlijn) in een volgende sessie af te ronden zonder helemaal opnieuw te moeten beginnen. Dit is trouwens alleen bij ZModem mogelijk.

We kunnen in onze club van geluk spreken dat er meerdere personen zijn die zich inzetten om iets moois te maken voor anderen en dat dan vervolgens op het Bulletin Board System zetten zodat een ieder ander er ook profijt van heeft.

Maar terug te komen op de berichten die men naar anderen kan sturen via het Bulletin Board System.

Op de meeste Bulletin Board Systemen heeft men de mogelijkheid om locale berichten in te geven maar ook om Echo-mail berichten in te geven. De berichten uit de eerste categorie worden dus op een bepaald Bulletin Board System ingegeven en blijven daar dan ook staan. Bijvoorbeeld berichten, gericht aan de Sysop van het betreffende Bulletin Board System. Berichten ingegeven in de Echo-mail gebieden (area) kunnen soms zelf wereld-wijd verspreid worden. Op het Bulletin Board System van de KGN (Kim Gebruikers club Nederland) staan op het moment steeds zo rond de 2000 berichten, gericht aan Jan en alleman. Ook gericht aan niet leden/gebruikers van het Bulletin Board System. Het is namelijk zo dat er zich Netwerken in Bulletin Board System land bevinden die de berichten (messages) 's nachts met elkaar uitwisselen. Zo bestaat er b.v. een area (term voor gebied) Handel. In die area kan men zijn spulletjes te koop aanbieden of om het een of ander vragen. Doordat deze area gekoppeld is aan een Netwerk (in Bulletin Board System termen : aangesloten is op de Echo-Mail) gebeurt het volgende: Het Bulletin Board System heeft bijgehouden dat er een bericht in de area handel is ingegeven en gaat deze dan uit het bestand halen (of eigenlijk kopiëren want het origineel blijft natuurlijk gewoon staan) om door te geven aan het Netwerk knooppunt. Op dit knooppunt komen dus op een gegeven moment ook berichten van andere Bulletin Board Systemen. Bij binnenkomst van deze berichten 'pakketten' gaat het knooppunt de binnengekomen berichten sorteren en weer verder sturen naar andere Bulletin Board Systemen die b.v. ook een area Handel hebben. Zo gebeurt het dus dat een bericht, ingegeven op een bepaald Bulletin Board System, na 1 á 2 dagen over het hele netwerk verspreid is (indien de andere Bulletin Board Systemen ook aangesloten zijn op die area en bij het zelfde netwerk horen). Helaas gebeurt het ook nog dat bepaalde mensen een bericht op een willekeurig Bulletin Board System ingeven daarna uitloggen en op een ander Bulletin Board System het zelfde bericht nog eens ingeven in de zelfde area. Dit heeft dan dus tot gevolg dat het bericht na enkele dagen meerdere keren op de diverse Bulletin Board Systemen voorkomt. Houd hier dus rekening mee als je in de toekomst van Echo-Mail gebruik wilt gaan maken.

Puur ondeskundigheid. Echo-Mail bestaat al erg lang maar er is helaas nog niet genoeg reclame voor gemaakt. Een handleiding hoe het te gebruiken (Echo-Mail) bestaat wel, maar er wordt nog niet genoeg naar verwezen. Zo is bijvoorbeeld reclame maken in de Echo-Mail ten strengste verboden.

Jacques Banser

De HCC-dagen.

Vrijdag 24 november en zaterdag 25 november was het weer zo ver. De Hobby Computer Club organiseerde voor de zoveelste keer de HCC-dagen. Hele volksstammen verlegden hun werkterrein naar de Jaarbeurshallen in Utrecht. Onze verslaggever was ter plaatse en maakte de volgende reportage...

Het is vijf uur in de morgen als mijn wekker met een luid gepiep het begin van de zaterdag aankondigt. Voorzichtig steek ik een teen buiten mijn bed om het in de kamer heersende klimaat aan een diepgaand onderzoek te onderwerpen. Na geconstateerd te hebben dat de temperatuur zich zo rond het vriespunt zal bewegen kan ik niet anders dan mijn snel afkoelende teen weer onder de warme dekens terug te trekken. Maar helaas, de plicht roept. Als ik er nu niet uit kom, dan komt er geen artikeltje. En dat zal meneer de hoofdredacteur zeker niet op prijs stellen. En moeder de vrouw wil brood op de plank. Kiezen of delen dus... Het wordt delen, vrees ik. Na nog een half uur gewacht te hebben (de temperatuur blijft op het zelfde niveau steken) haast ik mij naar de badkamer. Zo kom ik dan toch nog op tijd in mijn autootje. Krabben, krabben en nog eens krabben. Bah, ik haat de HCC. Niet omdat ze toevallig de HCC zijn, maar omdat ze de beurs niet dichterbij in de buurt georganiseerd hebben. De Twentehallen zijn er bij uitstek geschikt voor en de metro (oh ja?) kan mij zonder problemen op de bewuste lokatie brengen...

Het is toch al weer ruim tien uur geweest als ik de files na Hoevelaken gepasseerd ben. Over het stukje van dit beruchte knooppunt tot aan Utrecht doe ik normaliter een kwartiertje, maar al die computerfreaks op de weg vertragen de gang van mijn autootje danig. Het is ruim elf uur geweest als ik dan toch eindelijk de parkeerplaats van de jaarbeurs hallen opdraai.

De stroom mensen die zich langzaam richting ingang beweegt is al behoorlijk compact. Het verbaast me dat er nu al mensen terugkomen van de beurs, volgepakt met grote dozen waarop de drie bekende letters van de grootste computerfabrikant ter wereld prijken. "Made in Taiwan" vermeldt de doos die op nogal hardhandige wijze kennis maakt met de vloer. Zo te zien bevat het ding een monitor; ik ben benieuwd of de trotse eigenaar ooit nog het genoegen zal smaken naar zijn nieuwe aanwinst te turen...

Eenmaal binnen haast ik mij naar het restaurant. Op dit uur van de dag zitten er gelukkig nog niet veel mensen. Alleen een enkele vrouw die overduidelijk zit te wachten op de terugkeer van haar man. "Gelukkig accepteren ze hier geen cheques", denkt ze nog. Ik laat me de eerste kop koffie van de dag goed smaken en stort me vervolgens in het gewoel. Links en rechts worden mijn oren doof geschreeuwd door woeste verkopers die proberen hun waren aan de man te brengen. Dat ze daarbij zo af en toe wel eens hun buurman voor "achterlijke PDP-11" uitmaken mag de pret niet drukken.

In de verenigingshoek is het gelukkig een stuk rustiger. Tactisch opgesteld in een hoek probeert een catering firma nog wat gekoelde dranken te verkopen

De temperatuur in de hal schijnt omgekeerd evenredig te stijgen met de prijs van de blikjes cola.

Bij de stand van de Hack-Tic is het een drukte van belang. Een arme programmeur probeert zich met de moed der wanhoop te verdedigen tegen de woordenstroom van de in blauwe sweaters geklede freaks. Het witte "Hack-Tic" schreeuwt je van verre tegemoet. Als uiteindelijk blijkt dat het grootste deel van het publiek kiest voor de underdog positie van de sjofel aandoende technicus laten de heren zich niet van hun beste kant zien. Het "Ik zal wel eens een virus schrijven dat zich alleen via jouw programmatuur verspreidt" schalt door de hoge hal.

In een hoekje weggedrukt vind ik nog de schamele resten van wat de o zo trotse stand had moeten zijn van mijn geliefde club... Onze tijdelijke burens hadden hun expansiedrift waarschijnlijk gebotvierd op de stand. Nou ja, niet geheel ten onrechte, want vrijdag hadden we het mooi laten afweten... geen mankracht genoeg om de stand te kunnen bemannen, zo vernam ik uit betrouwbare bron.

In een hoekje weggedrukt vind ik nog de schamele resten van wat de o zo trotse stand had moeten zijn van mijn geliefde club... Onze tijdelijke burens hadden hun expansiedrift waarschijnlijk gebotvierd op de stand. Nou ja, niet geheel ten onrechte, want vrijdag hadden we het mooi laten afweten... geen mankracht genoeg om de stand te kunnen bemannen, zo vernam ik uit betrouwbare bron.

Een lichtkrant die aan een andere kraam hangt vertelt mij dat het al weer tegen sluitingstijd begint te lopen. De drukte neemt af, en een soort menselijke stroom begeeft zich richting uitgang. Alweer een geslaagde beursdag ten einde...

U.R. Frend.

De IBM-PC en z'n klonen (Deel 6).

De vorige keer hebben we gekeken naar de uitvoer van PC naar de gebruiker toe via een monitor en de mogelijke displayadapterkaarten. In deze aflevering gaan we eens kijken naar het tegenovergestelde: de invoer van data. Het meest gebruiken we daarvoor het wel bekende toetsenbord. IBM heeft voor de PC een oplossing voor het toetsenbord bedacht die eigenlijk best slim genoemd mag worden. Met een minimum aan verbindingen tussen de computer en het toetsenbord wordt een maximum aan mogelijkheden geschapen. De PC werd uitgerust met een toetsenbord waarvan de indeling zelfs een bepaalde standaard voldoet, ofschoon lang niet iedereen te spreken was over die indeling. Maar eerst de techniek van het toetsenbord.

6.1. De verbinding toetsenbord-computer.

Het toetsenbord wordt met het moederbord van de PC(/XT) verbonden via een gespiraliseerde vijf-aderige kabel, die eindigt in een normale vijf-polige DIN audio plug. In de kabel vinden we de volgende signalen terug:

GND	- pin 4
+ 5 Volt	- pin 5
Reset	- pin 3
KB-Clock	- pin 1
KB-Data	- pin 2

(Let op, de pinnen in de DIN plug zijn niet oplopend genummerd!).

Het toetsenbord krijgt zijn voeding van de 5 Volt van de PC. Ook het reset-sigitaal is gewoon verbonden met hetzelfde signaal waarmee in de PC bijvoorbeeld de CPU gereset wordt. Daarmee hebben we al drie van de vijf lijnen besproken.

De andere twee lijnen zijn bi-directioneel en voeren TTL-signalen. Ze zijn uitgevoerd als open-collector lijnen, zodat zowel de computer als het toetsenbord zelf de lijnen naar beneden kunnen trekken. De pull-up weerstanden bevinden zich zowel in de computer als in het toetsenbord aan beide einden van de kabel.

In normaal bedrijf is het toetsenbord de sturende bron en leest de computer de signalen die het toetsenbord verzendt. De data die door het toetsenbord wordt gegenereerd, wordt serieel over de KB-Data-lijn verzonden. Het betreft steeds bytes van 8 bits, het MS-bit eerst (Bij RS-232 wordt het LS-bit het eerst verstuurd). Iedere keer als er een bit op de KB-lijn staat, trekt het toetsenbord de KB-Clock-lijn naar beneden, waarop het moederbord het bit in-cloct in een 8-bit schuifregister met paralleluitgang. Zijn er 8 bits verzonden, dan verschijnt er een sig-

naal uit het schuifregister dat een IRQ1 genereert, terwijl tevens de KB-Data lijn wordt laag getrokken door het moederbord. Dit laatste is een signaal aan het toetsenbord dat er geen data verzonden kan worden. De processor kan nu de verzonden data lezen door poort A van de 8255, die verbonden is met de paralleluitgang van het schuifregister, uit te lezen. Vervolgens wordt een lijn in poort B (bit 7) van de 8255 even hoog gemaakt, waardoor het schuifregister gereset wordt en zich vult met nullen, en de interrupt verdwijnt. Met het verdwijnen van het interrupt signaal geeft het moederbord de KB-Data-lijn weer vrij. Het toetsenbord kan nu eventueel het volgende byte gaan versturen.

In principe is dit het protocol dat zich afspeelt tussen de computer en het toetsenbord. In theorie is het ook mogelijk dat het moederbord op dezelfde wijze data verstuurt naar het toetsenbord, maar hiervan wordt geen gebruik gemaakt in de PC(/XT). In de AT is wel echt twee-richting verkeer mogelijk.

De enige mogelijkheid die wel benut wordt door het moederbord is het activeren van een keyboard self-test. Het toetsenbord is namelijk uitgerust met een single-chip microcomputer (meestal een 8048 van intel). Als het moederbord gedurende tenminste 20 milliseconden zowel de clock- als data-lijn laag houdt, voert de 8048 de selftest uit. Er wordt onder andere gekeken of de checksum van de ROM-inhoud klopt, of het locale RAM werkt, en of er geen toetsen vastzitten. Verloopt de test goed, dan stuurt het toetsenbord de code \$AA naar het moederbord, anders de code \$FF.

6.2. De toets-indeling.

Het toetsenbord van de PC(/XT) heeft drie aparte gedeeltes wat betreft de soorten toetsen aangaat. Geheel links bevinden zich een tiental functietoetsen, gemerkt F1 t/m F10. Deze zijn bedoeld voor speciale functies en/of als 1-toets-commando's. In BASIC bijvoorbeeld zitten onder de tien toetsen een aantal veel gebruikte strings, zoals RUN <Enter> en LIST.

In het midden het grootste en ook meest vertrouwde deel: het schrijfmachinedeel. Hier vinden we de gebruikelijke verzameling toetsen voor het typen van het alfabet, de 10 cijfers en de complete ASCII-set aan leestekens en bijzondere tekens. Dit deel van het toetsenbord wordt gecompleteerd door een tweetal shift-toetsen aan weerszijden, een control toets en een CapsLock toets, waarmee we (in tegenstelling tot een schrijfmachine) de functie van de shift-toetsen voor de letters kunnen omkeren. Een

nieuwe toets is de Alt-toets, die zoals we verderop zullen zien, behalve als extra shift-toets ook nog een speciale functie heeft.

Helemaal rechts vinden we het zogenaamde numerieke deel van het toetsenbord: een extra set cijferttoetsen, die ook cursorbewegingen kunnen genereren. Verder in dit deel een plus- en een min-toets. Tenslotte een tweetal lock-toetsen: NumLock en ScrollLock. Met de NumLock kunnen we het numerieke deel cijfers laten genereren, met NumLock uit worden dit cursorbewegingen. De Scroll Lock-toets heeft geen speciale effecten op het toetsenbord tot gevolg. Ze is bedoeld voor softwarepakketten die hiermee de regel waarop de cursor zich bevindt op een vaste positie kunnen houden (Scroll Lock aan), danwel de cursor over het gehele scherm kunnen laten gaan (Scroll Lock uit). In het eerste geval zou het scherm bij iedere cursorbeweging in de Y-richting scrollen, in het tweede geval wordt er slechts gescrolled als de cursor buiten het scherm dreigt te raken. Er zijn echter maar weinig pakketten die deze functiemogelijkheid ook inderdaad benutten.

Caps- Num- en Scroll Lock zijn toggle-toetsen: eenmaal drukken is aan, een tweede keer drukken is weer uit. IBM maakte het de gebruiker van haar PC(XT) niet gemakkelijk: lampjes om de Lock-toestand aan te geven ontbraken (mede een gevolg van de 1-richting communicatie tussen moederbord en toetsenbord). De meeste klonen-toetsenborden bezitten wel lampjes in of bij de Lock-toetsen.

De beide shift-, de control-, en de Alt-toets(en) zijn actief zolang ze ingedrukt gehouden worden. Over ingedrukt houden gesproken: alle toetsen die een karakter genereren repeteren automatisch. Deze repeteerfunctie wordt door het toetsenbord zelf verzorgd.

De oplettende gebruiker heeft al ontdekt dat er toetsen zijn te vinden op het toetsenbord, waarvoor geen ASCII-codes bestaan. Ook was er in de inleiding beloofd, dat IBM een zo flexibel mogelijk systeem had bedacht voor het toetsenbord. Het toetsenbord stuurt dus geen ASCII-codes naar het moederbord, maar wat anders. Dat zijn:

6.3. Scancodes.

Het principe van de scancodes is zeer eenvoudig. Iedere toets op het toetsenbord heeft een nummer toegewezen gekregen. Deze nummers lopen van 1 (de Esc-toets) tot en met 83 (de Del-toets). Scancode nul wordt niet gebruikt. De soort toets maakt ook niet uit. Zo genereren ook de control-, Alt- en shift-toetsen gewoon scancodes. Drukken we een toets in,

dan zal het toetsenbord de betreffende scancode naar de PC sturen. Laten we de toets weer los, dan stuurt het toetsenbord wederom dezelfde scancode, maar nu met het MS-bit gezet. Op deze manier kan de PC dus alle toetsen herkennen, terwijl er ook bekend is welke toetsen ingedrukt zijn, en welke niet.

De repeteerfunctie wordt zoals gezegd door het toetsenbord zelf gerealiseerd: bij het vasthouden van een toets gaat het toetsenbord na verloop van tijd uit zichzelf de maak- en breekcodes van die toets versturen. Het toetsenbord kan meerdere ingedrukte toetsen herkennen: zowel de maak-codes als de breek-codes worden in dezelfde volgorde verzonden als waarin de toetsen bediend worden.

Het systeem-BIOS verzorgt de vertaling van de scancodes naar ASCII-teken, en plaatst deze, samen met de scancode in de toetsenbordbuffer, die 16 aanslagen (van twee bytes) diep is. Het BIOS houdt dus ook bij of er sprake is van shift, control of Alt-functies, of zelfs combinaties hiervan. Verder maakt het BIOS nog onderscheid tussen de linker en de rechter shift-toets, die aparte vlaggen hebben. In de praktijk ontvangen beide shift-toetsen dezelfde behandeling waardoor de gebruiker geen verschil merkt.

De truc met de scancodes en de aparte huishouding rond de shift-, control- en Alt-toetsen heeft tot gevolg dat het systeem ook niet-teken genererende combinaties als shift-control-M en dergelijke kan detecteren. Een aantal van deze bijzondere combinaties heeft dan ook een speciale functie gekregen, hieronder een lijstje:

Ctrl-Alt-Del: re-boot, warme start

Ctrl-ScrollLock: Break (ongeveer ^C)

Ctrl-Home: clear screen

Ctrl-NumLock: wacht op een toets

De laatste combinatie is bedoeld om de PC tijdelijk stil te kunnen zetten, bijvoorbeeld wanneer een grote directory wordt opgevraagd. Het BIOS wacht dan in een loop op een volgende toets. Zowel de sequentie Ctrl-NumLock als de volgende toets worden niet aan het gebruikersprogramma doorgegeven, zodat dit niet verstoord wordt door het stilzetten.

Een groot aantal BIOSsen voor turbo moederborden gebruiken de sequentie Ctrl-Alt-Numerieke min om de machine van clocksnelheid te laten veranderen.

Behalve het creëren van een extra shift-functie heeft de Alt-toets nog een tweede taak: het genereren van ASCII-codes die niet op het toetsenbord voorkomen. De IBM-ASCII-set bevat namelijk niet de ge-

bruikelijke 128 tekens, maar de volle 256 stuks. Alles beneden 128 kan gewoon of met de Ctrl- danwel een shift-toets worden gegenereerd. Tekens met ASCII-waarden boven 128 kunnen echter niet rechtstreeks worden ingetypt, en hiervoor is de Alt-toets. Stel we willen het teken met ASCII-waarde 152 invoeren. Druk hiertoe de Alt-toets in, en houdt deze vast. Typ nu op het numerieke deel achtereenvolgens een 1, een 5 en een 2, waarna de Alt-toets losgelaten wordt. Op het moment dat dat gebeurt, wordt de ASCII-waarde 152 in de toetsenbordbuffer gezet. Als de PC op dat moment de toetsenbordinvoer ook op het scherm laat zien, zal er een ÿ op het scherm verschenen zijn. Op deze manier kan dus iedere willekeurige ASCII-waarde worden ingevoerd. De laatste 3 cijfers zijn in principe geldig. Bij deze manier van invoeren maakt de stand van de NumLock niet uit: het BIOS leest immers scancodes.

De scancodes hebben nog een tweede voordeel: het maakt niet meer uit wat de indeling van het toetsenbord is, dezelfde toetsen blijven altijd dezelfde scan-code genereren. Dat is de reden achter de reeks KEYB-programma's die de toetsindeling aanpassen aan verschillende landen zoals Duitsland (QWERTZ, met umlaut-tekens) of Frankrijk (AZERTY, met accent-tekens). Het toetsenbord is dus vrij programmeerbaar: je hoeft alleen maar een nieuwe scancode-naar-ASCII tabel te definiëren.

6.4. Nadere beschrijvingen van de bijzondere toetsen.

Een aantal toetsen verdienen een nadere uitleg. De meeste bijzondere toetsen bevinden zich in het numerieke deel van het toetsenbord. De functie van de pijltjestoetsen zal duidelijk zijn: zij bewegen de cursor. Overigens is het zo, dat wanneer NumLock aan staat, de andere functies in het numerieke deel gebruikt kunnen worden door 1 van de shift-toetsen ingedrukt te houden. NumLock doet dus hetzelfde met het numerieke deel wat CapsLock voor de letters doet: de functie van de shift-toetsen omkeren.

Behalve de cursortoetsen, zijn ook de overige speciale functies in het numerieke deel logisch ingedeeld. Zo horen 7/Home en 1/End bij elkaar: Home springt naar de linker bovenhoek van het scherm, en End naar de linker onderhoek (sommige pakketten springen naar het laatste karakter op de onderste schermregel). De toetsen 9/PgUp en 3/PgDn zijn bedoeld als blader-toetsen: de bedoeling is om een scherm voorwaarts (PgUp), danwel een scherm terug (PgDn) te springen. Sommige tekstverwerkers (zoals WordPerfect) gebruiken niet het scherm als bladergrootte, maar de paginaindeling van het document als maatstaf, erg hinderlijk.

De nul-toets bevat de functie Ins, voor Insert. In nagenoeg alle toepassingen zorgt de Ins-toets ervoor, dat de invoeg-mode wordt in- respectievelijk uitgeschakeld. Het is dus een toggle-functie. De Del-toets is bedoeld om het karakter onder de cursor te wissen. (De backspace is in de IBM-wereld het karakter dat het karakter links van de cursor wist). De numerieke min en de numerieke plus genereren separate scancodes, kunnen dus eventueel ook voor speciale functies worden ingezet. Sommige pakketten maken hier ook gebruik van.

6.5. Lijst van scancodes.

Wellicht wat saai, maar misschien toch handig:

Toets	Scancode	Toets	Scancode
Esc	1	\\	43
1/!	2	Z	44
2/@	3	X	45
3/#	4	C	46
4/\$	5	V	47
5/%	6	B	48
6/^	7	N	49
7/&	8	M	50
8/*	9	,/	51
9/(10	/	52
0/)	11	//?	53
-/_	12	R.shift	54
=/+	13	*/PrtScr	55
Backspace	14	Alt	56
TAB	15	Spatie	57
Q	16	CapsLock	58
W	17	F1	59
E	18	F2	60
R	19	F3	61
T	20	F4	62
Y	21	F5	63
U	22	F6	64
I	23	F7	65
O	24	F8	66
P	25	F9	67
[/{	26	F10	68
]/*	27	NumLock	69
Enter	28	ScrollLock	70
Ctrl	29	Home/7	71
A	30	Up/8	72
S	31	PgUp/9	73
D	32	Numeric -	74
F	33	Left/4	75
G	34	5	76
H	35	Right/6	77
J	36	Numeric +	78
K	37	End/1	79
L	38	Down/2	80
;/:	39	PgDn/3	81
'/"	40	Ins/0	82
~/~	41	Del/.	83
L.shift	42		

(Alle scancodes in decimaal!).

6.6. Andere toetsenborden.

Al spoedig kwam er kritiek op het toetsenbord van de IBM-PC. Niet omdat het gammel was, want een echt IBM toetsenbord kun je maar beter niet op je tenen krijgen: dat wordt een ritje ziekenhuis. De kritiek kwam op de indeling. De meeste mensen vonden de Return-toets (door IBM consequent de Enter-toets genoemd) te klein, en teveel ingesloten door de andere toetsen. Een tweede punt van kritiek was het ontbreken van de lampjes bij of in de Lock-toetsen.

Ook het numerieke deel was onderwerp van kritiek: mensen die gewend waren op die plek veel cijfers in te voeren zaten ineens zonder cursorbesturing als NumLock werd aangezet. Kortom, perfect was het niet helemaal.

De eerste antwoorden kwamen zoals meestal van de klonenbouwers. Het toevoegen van de lampjes bij de Lock-toetsen is niet zo'n toer, ofschoon het toetsenbord en de PC ieder voor zich bijhouden welke Locks er actief zijn. Als het moederbord een scan-code mist van een Lock-toets, dan klopt de indicatie niet meer....

Met het verschijnen van PC/AT bracht IBM ook een tweetal nieuwe toetsenborden uit, een klein dat leek op het PC/XT toetsenbord, en een groter model, dat een apart cursor- en een apart numeriek deel heeft. Dit laatste toetsenbord gaat door het leven als 'Enhanced keyboard' en heeft 101 toetsen tegen 83 voor het PC(/XT) toetsenbord. Het kleine AT-toetsenbord heeft 1 toets meer dan het XT toetsenbord: 84 stuks dus.

Ofschoon de indeling van het XT-toetsenbord voldoet aan een bepaalde DIN-standaard, werd de standaard voor de nieuwe toetsenborden door IBM verlaten. Zo zit op het 84-key AT-toetsenbord de Esc-toets op een plek waar hij beslist niet hoort: links boven in het numerieke deel.

Beide nieuwe toetsenborden hadden een lekker grote Enter-toets (het grote toetsenbord heeft er zelfs twee: ook 1 in het numerieke deel). IBM zou IBM echter niet zijn, als de AT-toetsenborden op de PC(/XT) zouden werken. Dat doen ze dus niet: het

scancodetransmissieformaat werd uitgebreid met een parity check, terwijl het toetsenbord ook commando's van de PC kan ontvangen.

Ook op dit probleem werd alert door de Oosterse klonenbouwers gereageerd: er verschenen omschakelbare toetsenborden die zowel op XT's als op AT's gebruikt konden worden. Beide AT-toetsenborden verschenen in omschakelbare vorm, ofschoon de Enhanced uitvoering tegenwoordig verreweg het populairst is.

Niet alle XT-BIOSsen weten goed raad met zo'n omschakelbaar Enhanced toetsenbord. Zo'n toetsenbord heeft bijvoorbeeld een aparte Print_Screen toets, een functie die op een normaal XT-toetsenbord een shift-functie is. Om deze toets op een XT te kunnen gebruiken, sturen de meeste Enhanced toetsenborden de volgende reeks scancodes naar de PC:

```
linker shift-toets neer
Scancode 55 (* /PrtScr op XT-kb)
Scancode 55 + 128
linker shift-toets los
```

Het toetsenbord onthoudt hierbij ook nog, of er een shift-, control- of Alt-toets ingedrukt was. Als dat zo was, stuurt het eerst ook nog de loslaat-scancode voor die toets naar de computer. Na het loslaten van de Print_Screen toets volgt dan nog de maak-scancode voor die shift-toets. Kortom, in sommige gevallen krijgt de computer tot zes scancodes binnen voor 1 toets, in tempo bepaald door het toetsenbord, terwijl dit op de PC er hooguit twee zijn, beide ingedrukt door de (langzame) gebruiker. Niet alle BIOSsen weten daar raad mee.....

6.7 De volgende keer.....

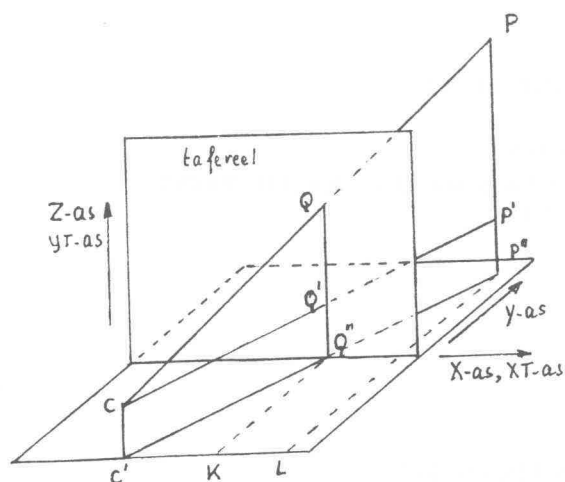
Wordt het echt leuk: dan is het eindelijk tijd om eens naar het zenuwcentrum van de PC te gaan kijken: het systeem-BIOS. We zullen zien dat het BIOS niet alleen druk is om de machine overeind te krijgen na een power-up, maar ook een groot aantal handige functies voor ons kan uitvoeren.

Tot dan.

Nico de Vries

STEREOMETRISCHE FIGUREN OP DE COMPUTER.

Voor een zo natuurgetrouw mogelijke weergave van een ruimtelijke figuur wordt gebruik gemaakt van de perspectief. Achter een (meestal) verticaal vlak (het tafereel) wordt het te projecteren voorwerp geplaatst, terwijl zich voor dit vlak het centrum (het oog) bevindt. We gaan voor het punt $P(XP, YP, ZP)$ de formules afleiden, waarmee op ons vlakke beeldscherm een perspectiefisch beeld kan worden verkregen. Het linkeroog bevindt zich in punt $C(C1-OA, C2, C3)$, waarin $2 \cdot OA$ de afstand tussen beide ogen is. We denken dat het beeldscherm voorzien is van een assenstelsel met het centrum van het beeldscherm als oorsprong en de cm als eenheid. De coördinaten op het beeldscherm geven we aan met XT, YT . Volledigheidshalve is nog vermeld dat alle getekende lijnen loodrecht staan op een van de getekende vlakken behalve CP, CP' en $C'P$.



In de figuur zien we 2 paar gelijkvormige driehoeken. Uit het ene volgt de evenredigheid:

$PP' : QQ' = CP' : CQ' = C'P' : C'Q'$ en uit het andere stel: $C'P' : C'Q' = P''L : Q''K = C'L : C'K$

Samengevoegd levert dit: $PP' : QQ' = P''L : Q''K$ (1) en ook $C'L : C'K = P''L : Q''K$ (2)

Invullen van de gegevens geeft voor deze evenredigheden:

$$(ZP-C3):(YT-C3) = (YP+C2):C2 \quad (1a)$$

$$(XP-C1+OA):(XT-C1+OA) = (YP+C2):C2 \quad (2a)$$

Na enige herleiding (waarbij we tevens (XT, YT) vervangen door (XL, YL) voor linkeroog) krijgen we de formules:

$$XL = \frac{YP \cdot (C1-OA) + XP \cdot C2}{YP + C2} \quad (1b) \text{ en}$$

$$YL = \frac{ZP \cdot C2 - YP \cdot C2}{YP + C2} \quad (2b)$$

Voor het rechteroog krijgen we de formules door OA te vervangen door $-OA$. Overgang op beeldschermcoördinaten (XS, YS) geeft: $XS = (4 + XP) \cdot 23$ en $YS = (21 - YP) \cdot 19$ met $(14, 21)$ de coördinaten van het beeldscherm midden, de getallen 23 en 19 geven het aantal pixels/cm. Door substitutie van 1a en 1b hierin vinden we de formules uit subroutine 20000 van het programma.

Het ruimtelijk effect wordt verkregen m.b.v. een rood/groene bril, groen glas voor het linkeroog en rood voor het rechter.

Wellicht ten overvloede zij vermeld dat het ruimtelijk effect alleen bereikt wordt met een kleurenmonitor.

A.P.OERLEMANS

```

10  CLS:KEY OFF:SCREEN 0,0,0
20  PRINT TAB(20)"*****"
30  PRINT TAB(20)"*
40  PRINT TAB(20)"* RUIMTELIJKE FIGUREN *"
50  PRINT TAB(20)"*
60  PRINT TAB(20)"*****":PRINT
70  FACTOR = ATN(1)/45:REM Omrekenfactor radialen/graden
80  OA = 3:C2 = -40:REM OA = halve oogafstand.C2 = afstand oog-tafereel.
100 DIM X(7),Y(7),Z(7),XL(7),YL(7),XR(7),YR(7),STR(7),HK(7)
500 REM = = = = =
510 REM = keuze =
520 REM = = = = =
530 PRINT "Maak een keuze uit:"

```

```

540 PRINT "1. tetraeder"
550 PRINT "2. kubus"
560 PRINT "3. bol"
570 PRINT "4. cilinder"
580 PRINT "5. kegel"
590 PRINT "6. tetraeder met in-bol"
600 PRINT "7. kubus met in-bol"
610 PRINT "8. schroefvlak"
620 PRINT "9. schroefdraad"
630 PRINT "S. stoppen."
640 PRINT :PRINT
650 PRINT "Met de < Return > -toets komt men vanuit de figuren terug in het menu."
660 PRINT
900 PRINT "Type de keuze in: ";
910 INPUT KEUZE$:IF KEUZE$ = "S" OR KEUZE$ = "s" THEN CLS:END
920 KEUZE = VAL(KEUZE$):IF KEUZE OR KEUZE9 THEN 910
930 CLS
940 ON KEUZE GOSUB 1000,2000,3000,4000,5000,6000,7000,8000,9000
950 INPUT A$
960 SCREEN 0,0,0:
970 LOCATE 7:GOTO 500
999 END
1000 REM *****
1010 PRINT TAB(25) "VIERVLAK: GEGEVENS ":PRINT :PRINT
1020 REM *****
1030 INPUT "Hoe groot is de lengte van de zijden";ZIJDE:PRINT
1040 INPUT "Over welke hoek(in het hor. vlak) moet het viervlak gedraaid worden";PHI:PRINT
1050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
1060 INPUT "Op welke breedte bevindt zich het oog";C1
1070 PHI = PHI*FACTOR :REM Van radialen naar graden
1080 GOSUB 1500:RETURN
1500 REM *****
1510 REM * VIERVLAK: TEKENEN *
1520 REM *****
1530 SCREEN 3:COLOR 7,1
1540 X(0) = -ZIJDE*SIN(PHI)/SQR(3):Y(0) = ZIJDE*COS(PHI)/SQR(3)
1550 SQ = SQR(.75):X(1) = -X(0)/2 - Y(0)*SQ:Y(1) = -Y(0)/2 + X(0)*SQ
1560 X(2) = -X(0)/2 + Y(0)*SQ:Y(2) = -Y(0)/2 - X(0)*SQ:X(3) = 0:Y(3) = 0
1570 Z(0) = -ZIJDE/SQR(24):Z(1) = Z(0):Z(2) = Z(0):Z(3) = -3*Z(0)
1580 FOR I = 0 TO 3
1590 XP = X(I):YP = Y(I):ZP = Z(I):GOSUB 2000
1600 XL(I) = XSL:YL(I) = YSL:XR(I) = XSR:YR(I) = YSR
1610 NEXT I
1620 LINE (XL(0),YL(0)) - (XL(1),YL(1)),4:LINE -(XL(2),YL(2)),4
1630 LINE -(XL(0),YL(0)),4:LINE -(XL(3),YL(3)),4:LINE-(XL(1),YL(1)),4
1640 LINE (XL(2),YL(2))-(XL(3),YL(3)),4
1650 LINE (XR(0),YR(0)) - (XR(1),YR(1)),5:LINE -(XR(2),YR(2)),5
1660 LINE -(XR(0),YR(0)),5:LINE -(XR(3),YR(3)),5:LINE-(XR(1),YR(1)),5
1670 LINE (XR(2),YR(2))-(XR(3),YR(3)),5
1680 RETURN
2000 REM *****
2010 PRINT TAB(25)"KUBUS: GEGEVENS ":PRINT:PRINT
2020 REM *****
2030 INPUT "Hoe groot is de lengte van de ribben";RIBBE:PRINT
2040 INPUT "Over welke hoek (in het hor. vlak) moet de kubus gedraaid worden";PHI:PRINT
2050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
2060 INPUT "Op welke breedte bevindt zich het oog";C1
2070 PHI = PHI*FACTOR:REM Van radialen naar graden
2080 GOSUB 2500:RETURN

```

```

2500 REM *****
2510 REM * KUBUS: TEKENEN *
2520 REM *****
2530 SCREEN 3:COLOR 7,1
2540 X(0) = (COS(PHI) + SIN(PHI))*RIBBE/2:Y(0) = (SIN(PHI)-COS(PHI))*RIBBE/2:Z(0) = -RIBBE/2
2550 X(4) = X(0):Y(4) = Y(0):Z(4) = RIBBE/2
2560 FOR I = 1 TO 3
2570 X(I) = -Y(I-1):Y(I) = X(I-1):Z(I) = -RIBBE/2
2580 X(I+4) = X(I):Y(I+4) = Y(I):Z(I+4) = RIBBE/2
2590 NEXT I
2600 FOR I = 0 TO 7
2610 XP = X(I):YP = Y(I):ZP = Z(I):GOSUB 20000
2620 XL(I) = XSL:YL(I) = YSL:XR(I) = XSR:YR(I) = YSR
2630 NEXT I
2640 LINE (XL(0),YL(0)) - (XL(1),YL(1)),4:LINE -(XL(2),YL(2)),4
2650 LINE -(XL(3),YL(3)),4:LINE -(XL(0),YL(0)),4:LINE -(XL(4),YL(4)),4
2660 LINE -(XL(5),YL(5)),4:LINE -(XL(6),YL(6)),4:LINE -(XL(7),YL(7)),4
2670 LINE -(XL(4),YL(4)),4:LINE (XL(1),YL(1))-(XL(5),YL(5)),4
2680 LINE (XL(2),YL(2))-(XL(6),YL(6)),4:LINE(XL(3),YL(3))-(XL(7),YL(7)),4
2690 LINE (XR(0),YR(0))-(XR(1),YR(1)),5:LINE -(XR(2),YR(2)),5
2700 LINE -(XR(3),YR(3)),5:LINE -(XR(0),YR(0)),5:LINE -(XR(4),YR(4)),5
2710 LINE -(XR(5),YR(5)),5:LINE -(XR(6),YR(6)),5:LINE -(XR(7),YR(7)),5
2720 LINE -(XR(4),YR(4)),5:LINE (XR(1),YR(1))-(XR(5),YR(5)),5
2730 LINE (XR(2),YR(2))-(XR(6),YR(6)),5:LINE(XR(3),YR(3))-(XR(7),YR(7)),5
2740 RETURN
3000 REM *****
3010 PRINT TAB(22)"* BOL: GEGEVENS *":PRINT:PRINT
3020 REM *****
3030 INPUT "Hoe groot is de straal";RADIUS:PRINT
3040 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
3050 INPUT "Op welke breedte bevindt zich het oog";C1
3060 GOSUB 3500:RETURN
3500 REM *****
3510 REM * BOL: TEKENEN *
3520 REM *****
3530 SCREEN 3:COLOR 7,1
3540 XP = 0:ZP = 0
3550 FOR I = 0 TO 6
3560 YP = RADIUS*SIN((I-3)*22.5*FACTOR):GOSUB 20000
3570 XL(I) = XSL:XR(I) = XSR:YL(I) = YSL:YR(I) = YSR
3580 STR(I) = C2*RADIUS*COS((I-3)*22.5*FACTOR)/(YP + C2)*23
3590 NEXT I
3600 FOR I = 0 TO 6
3610 CIRCLE (XL(I),YL(I)),STR(I),4
3620 CIRCLE (XR(I),YR(I)),STR(I),5
3630 NEXT I
3640 YP = RADIUS:GOSUB 20000:CIRCLE(XSL,YSL),2,4:CIRCLE(XSR,YSR),2,5
3650 YP = -RADIUS:GOSUB 20000:CIRCLE(XSL,YSL),2,4:CIRCLE(XSR,YSR),2,5
3660 RETURN
4000 REM *****
4010 PRINT TAB(22)"* CILINDER: GEGEVENS *":PRINT:PRINT
4020 REM *****
4030 INPUT "Wat is de hoogte van de cilinder";HC:PRINT
4040 INPUT "Hoe groot is de straal van de cirkel";RC:PRINT
4050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
4060 INPUT "Op welke breedte bevindt zich het oog";C1
4070 GOSUB 4500:RETURN
4500 REM *****
4510 REM * CILINDER: TEKENEN *

```

```

4520 REM *****
4530 SCREEN 3:COLOR 7,1
4540 ZP = HC/2:XP = -RC:YP = 0:GOSUB 20000
4550 XLA = XSL:YLA = YSL:XRA = XSR:YRA = YSL
4560 ZP = -HC/2:GOSUB 20000
4570 XLB = XSL:YLB = YSL:XRB = XSR:YRB = YSL
4580 FOR I = -180 TO 180 STEP 5
4590 ZP = HC/2:XP = RC*COS(I*FACTOR):YP = RC*SIN(I*FACTOR):GOSUB 20000
4600 XL1 = XSL:YL1 = YSL:XR1 = XSR:YR1 = YSR
4610 ZP = -HC/2:GOSUB 20000
4620 XL2 = XSL:YL2 = YSL:XR2 = XSR:YR2 = YSR
4630 LINE (XL1,YL1)-(XL2,YL2),4:LINE (XR1,YR1)-(XR2,YR2),5
4640 LINE (XLA,YLA)-(XL1,YL1),4:LINE(XRA,YRA)-(XR1,YR1),5
4650 LINE (XLB,YLB)-(XL2,YL2),4:LINE(XRB,YRB)-(XR2,YR2),5
4660 XLA = XL1:YLA = YL1:XRA = XR1:YRA = YR1:XLB = XL2:YLB = YL2:XRB = XR2:YRB = YR2
4670 NEXT I
4680 RETURN
5000 REM *****
5010 PRINT TAB(22)"* KEGEL: GEGEVENS *":PRINT:PRINT
5020 REM *****
5030 INPUT "Wat is de hoogte van de kegel";HK:PRINT
5040 INPUT "Hoe groot is de straal van de cirkel";RK:PRINT
5050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
5060 INPUT "Op welke breedte bevindt zich het oog";C1
5070 GOSUB 5500:RETURN
5500 REM *****
5510 REM * KEGEL: TEKENEN *
5520 REM *****
5530 SCREEN 3:COLOR 7,1
5540 ZP = -HK/2:XP = -RK:YP = 0:GOSUB 20000
5550 XLA = XSL:YLA = YSL:XRA = XSR:YRA = YSL
5560 ZP = HK/2:XP = 0:GOSUB 20000
5570 XLB = XSL:YLB = YSL:XRB = XSR:YRB = YSL
5580 FOR I = -180 TO 180 STEP 15
5590 ZP = -HK/2:XP = RK*COS(I*FACTOR):YP = RK*SIN(I*FACTOR):GOSUB 20000
5600 XL1 = XSL:YL1 = YSL:XR1 = XSR:YR1 = YSR
5610 LINE (XL1,YL1)-(XLB,YLB),4:LINE (XR1,YR1)-(XRB,YRB),5
5620 LINE (XLA,YLA)-(XL1,YL1),4:LINE(XRA,YRA)-(XR1,YR1),5
5630 XLA = XL1:YLA = YL1:XRA = XR1:YRA = YR1
5640 NEXT I
5650 RETURN
6000 REM *****
6010 REM * TETRAEDER MET BOL *
6020 REM *****
6030 GOSUB 1000:REM GEGEVENS TETRAEDER
6040 RADIUS = -Z(0):REM Straal v.d. bol bepalen.
6050 GOSUB 1500:GOSUB 3500
6060 RETURN
7000 REM *****
7010 REM * KUBUS MET BOL *
7020 REM *****
7030 GOSUB 2000:REM GEGEVENS KUBUS
7040 RADIUS = RIBBE/2:REM Straal v.d. bol bepalen.
7050 GOSUB 2500:GOSUB 3500
7060 RETURN
8000 REM *****
8010 PRINT TAB(20)"* SCHROEFVLAK: GEGEVENS *":PRINT:PRINT
8020 REM *****
8030 INPUT "Hoe groot is de diameter van het schroefvlak";RS:PRINT

```

```

8040 INPUT "Hoeveel gangen heeft het schroefvlak";GANG:PRINT
8050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
8060 INPUT "Op welke breedte bevindt zich het oog";C1
8070 PHI = PHI*FACTOR:REM Van radialen naar graden
8080 GOSUB 8500:RETURN
8500 REM *****
8510 REM * SCHROEFVLAK: TEKENEN *
8520 REM *****
8530 SCREEN 3:COLOR 7,1
8540 FOR I=0 TO 360*GANG STEP 5
8550 XP=RS*COS(I*FACTOR):YP=RS*SIN(I*FACTOR):ZP=-4+I/(45*GANG):GOSUB 20000
8560 XWL=XSL:YWL=YSL:XWR=XSR:YWR=YSR
8570 XP=0:YP=0:GOSUB 20000
8580 XAL=XSL:YAL=YSL:XAR=XSR:YAR=YSR
8590 LINE (XWL,YWL)-(XAL,YAL),4:LINE (XWR,YWR)-(XAR,YAR),5
8600 XAL=XSL:YAL=YSL:XAR=XSR:YAR=YSR
8610 NEXT I
8630 RETURN
9000 REM *****
9010 PRINT TAB(25) "* SCHROEFLIJN GEGEVENS *":PRINT :PRINT
9020 REM *****
9030 INPUT "Hoe groot is de diameter van de schroefdraad";RS:PRINT
9040 INPUT "Hoeveel gangen heeft de schroefdraad";GANG:PRINT
9050 INPUT "Op welke hoogte bevindt zich het oog";C3:PRINT
9060 INPUT "Op welke breedte bevindt zich het oog";C1
9070 PHI = PHI*FACTOR:REM Van radialen naar graden
9080 GOSUB 9500:RETURN
9090 RETURN
9500 REM *****
9510 REM * SCHROEFLIJN TEKENEN *
9520 REM *****
9530 SCREEN 3:COLOR 7,1
9540 XP=RS:YP=0:ZP=-4:GOSUB 20000
9550 XAL=XSL:YAL=YSL:XAR=XSR:YAR=YSR
9560 FOR I=10 TO 360*GANG STEP 10
9570 XP=RS*COS(I*FACTOR):YP=RS*SIN(I*FACTOR):ZP=-4+I/(45*GANG):GOSUB 20000
9580 XWL=XSL:YWL=YSL:XWR=XSR:YWR=YSR
9590 LINE (XAL,YAL)-(XWL,YWL),4:LINE (XAR,YAR)-(XWR,YWR),5
9600 XAL=XWL:YAL=YWL:XAR=XWR:YAR=YWR
9610 NEXT I
20000 REM *****
20010 REM *
20020 REM * BEELDSCHERM COORDINATEN BEREKENEN *
20030 REM *
20040 REM *****
20060 XSR=(14+(C2*XP+(C1+OA)*YP)/(YP+C2))*23
20070 XSL=(14+(C2*XP+(C1-OA)*YP)/(YP+C2))*23
20080 YSR=(10-(C2*ZP+C3*YP)/(YP+C2))*19:YSL=YSR
20090 RETURN

```

